NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

ADAPTIVE WINDOWS VIA
KALMAN FILTERING
IN THE SPECTRAL DOMAIN

by

Ronald C. Adamo

March, 1991

Thesis Advisor: Ralph Hippenstiel

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.:
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (*If applicable*) AW	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000	7b. ADDRESS (*City, State, and ZIP Code*) Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (*If applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS Program Element No / Project No / Task No / Work Unit Accession Number

11. TITLE (*Include Security Classification*)
Adaptive Windows via Kalman Filtering in the Spectral Domain

12. PERSONAL AUTHOR(S) Adamo, Ronald Carl

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*year, month, day*)	15. PAGE COUNT
Master's Thesis	From To	1991 March	~~138~~ 137

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS (*continue on reverse if necessary and identify by block number*)
FIELD	GROUP	SUBGROUP	signal processing; Kalman filter; grey-tone displays; periodogram; spectral analysis

19. ABSTRACT (*continue on reverse if necessary and identify by block number*)

Application of classical windows to time series data is a means of enhancing the performance of the periodogram. Use of these classical windows results in the broadening of the spectral mainlobe. A Kalman filter will smooth spectral data by dividing the periodogram of unwindowed time series data into piecewise constant segments, ideally into noise-only and signal-only segments. This allows for a more accurate representation of the mainlobe of the original periodogram. The Kalman filter was modified to alter the filter parameter (beta) during filtering to provide maximum smoothing during the noise-only portion of the periodogram while leaving the main spectral peak essentially unaltered. A second modification was made to substitute the original raw values of the periodogram for the filter estimates during detected up-transitions while smoothing the noise-only segments in the same manner as in the original Kalman filter algorithm. This further enhances the preservation of the mainlobe of the periodogram and lowers the noise floor 1 to 3 dB over that of the original Kalman filter. These processes were further enhanced by stacking the output periodograms and displaying them as LOFAR output on the Sun workstation. NCAR graphics grey-toning is used to generate the LOFAR displays.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS REPORT ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Hippenstiel, R. D.	22b. TELEPHONE (*Include Area code*) 408-646-2633	22c. OFFICE SYMBOL EC/Hi

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Approved for public release; distribution is unlimited.

Adaptive Windows via Kalman Filtering
in the Spectral Domain

by

Ronald C. Adamo
Lieutenant Commander, United States Navy
B.S., Tulane University, 1980

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN APPLIED SCIENCE
(Antisubmarine Warfare)

from the

NAVAL POSTGRADUATE SCHOOL
March 1991

ABSTRACT

Application of classical windows to time series data is a means of enhancing the performance of the periodogram. Use of these classical windows results in the broadening of the spectral mainlobe. A Kalman filter will smooth spectral data by dividing the periodogram of unwindowed time series data into piecewise constant segments, ideally into noise-only and signal-only segments. This allows for a more accurate representation of the mainlobe of the original periodogram. The Kalman filter was modified to alter the filter parameter β during filtering to provide maximum smoothing during the noise-only segment and maximum sensitivity in the vicinity of the spectral peak of the periodogram. This modification results in a smoothing of the noise-only portion of the periodogram while leaving the main spectral peak essentially unaltered. A second modification was made to substitute the original raw values of the periodogram for the filter estimates during detected up-transitions while smoothing the noise-only segments in the same manner as in the original Kalman filter algorithm. This further enhances the preservation of the mainlobe of the periodogram and lowers the noise floor 1 to 3 dB over that of the original Kalman filter. These processes were further enhanced by stacking the output periodograms and displaying them as LOFAR output on the Sun workstation. NCAR graphics grey-toning subroutine is used to generate the LOFAR displays.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. MAN-MACHINE INTERFACE

As the computational powers of the computer continues to improve, the ability to process large amounts of data in relatively short periods of time has also improved. In addition it is now possible do much of these computations on desk-top computers in the office or even at home. In the past the results (i.e., output) of these computations were provided in the form of paper printout. The individual who must digest this information and make a decision or draw conclusions from this data was faced with a serious problem. The value of the information provided will depend on the ability of the decision maker to understand what is being presented.

Recent advances have made the computer very efficient at gathering and storing data. The problem is retrieving the data in a usable form and presenting it to the user (i.e., decision maker). [Ref. 1]

The use of computer graphical displays is an effective solution to this problem. Human information processing operations depend on the sensory reception of relative stimulation. The information we are interested in is contained in this stimuli. This stimuli may be coded in visual displays and reproduced. The reproductions can be

modified by enlargement, amplification, filtering or enhancement. When indirect sensing applies, there is a human factors aspect that must enter into the design of these displays. [Ref. 2].

The human brain has a vast capacity for non-verbal perception. This trait can be optimized since computers have the ability to graphically communicate up to 100,000 times more effectively than statistical printouts. It is estimated that the total capacity of the visual channel can be as high as 30-40 million bits of information per second. This is equivalent to 48-72 million words per minute, assuming seven bits per character and five characters per word. Since humans can not read this fast the effectiveness of graphics over alpha-numeric printouts is clear [Ref. 1:p.5].

Often people can sense many sources of information directly without much problem. There are however, many instances where direct sensing is not adequate. Some of these situations are mentioned below:

1. Information is at or below threshold values that may need to be amplified by some means (i.e., electronic, visually, etc...).
2. Information may be to large or massive and require reduction or consolidation.
3. Information could be embedded in noise that may need to be filtered out.
4. Stimuli may be of the type that could be better sensed if converted to some other type of sensory display (e.g., graphs or color coding to represent quantitative data).

Generally speaking, the information being presented by displays is either dynamic or static. Dynamic information is continually changing or may change with time. The randomness associated with the noise in which a signal may be present is considered dynamic. Displays may also present static information. This could be information that does not change with time of for information that remains constant for some fixed period of time. We are interested in a dynamic display in this discussion. For a detailed listing of specific examples of dynamic and static displays see [Ref. 2:p.49].

Selecting the proper display is very important and will depend on the type of data being evaluated. Some general guidelines for the selection of visual displays follows:

1. Use the simplest display concept that will most rapidly provide the user with the information needed to properly interpret the data of interest. The more sophisticated and complicated the display the longer it will take the user to digest and the more likely the user will be to misinterpret the data being displayed.

2. Use the least precise format needed to covey the desired information. Overemphasizing the accuracy of readout required can result in increasing the users response time, add to users fatigue or mental stress, and can ultimately cause unnecessary mistakes.

3. Use a display that is natural to the user. The picture the user sees should convey the expected interpretation that is associated with the data being displayed.

4. Use the most practical display technique for the expected viewing environment and operating viewing conditions. [Ref. 3].

Consider the case where the data of interest is a "n by n" matrix of considerable size and the user is interested in

locating the maximum value in that matrix while also recognizing any consistencies between the maximum values in each row. In addition, suppose that the exact numerical value of those maximum numbers are less important than their locations in their respective rows. It would be useful if the maximum value of the matrix could be assigned a visual parameter and all the other numerical values could be assigned a visual parameter that could be scaled relatively to that maximum. This visual coding takes up significantly less space than would be required when displaying the numerical value of each piece of data and would allow one look at the entire matrix at once. Consider a coding system in which the visual scale uses grey tones over the range of white to black. The largest value in the data set is coded to represent the darkest shade and the smallest value is assigned the absence of color (i.e. white). All values in between will be assigned grey-tones relative to their numerical standing in relationship to the maximum and minimum values. This is the basic idea behind the grey-tone (LOFAR) display utilized in this research.

B. RESEARCH OVERVIEW

The estimate of the spectral content of a segment of a process is most commonly accomplished by the use of a periodogram. The performance of the periodogram has significant shortcomings when dealing with the detection of

signals in noise. One of these shortcomings is the window function sidelobe effects that result from processing data sets of finite length. By using tapered window functions, such as the Hamming, the von Hann, or the Bartlett window, one can improve the performance of the periodogram by minimizing the effects caused by the discontinuities of the data at the boundaries [Ref. 4]. The periodogram also has a large variance (independent of data length used). Typically, the mean equals the standard deviation in the noise-only case. An other method of improving the performance of the periodogram is by averaging over a succession of sequential periodograms. The smoothing of the spectral estimate is due to the reduction of the variance of the estimate. Loss of resolution and broadening of the main spectral peaks almost always results from the use of either of these techniques. In the first case, the broadening is caused by the convolution of the window Fourier transform with the Fourier transform of the data, while in the second case the broadening potentially comes from having to shorten transform lengths. William Go [Ref. 5] performed research that examined the application of a Kalman filter to a rectangular windowed data set. In his work he was able to demonstrate that the Kalman filter could filter (i.e. smooth) the noise portions of the spectral estimate while leaving the main spectral peaks essentially unaltered. The results realized were preservation of the original periodograms resolution while accentuating the

dominant spectral peaks as they rise out of the noise floor. Go used single and multiple sinusoids in Gaussian white noise to evaluate the Kalman filter's performance in signal detection and resolution at different input signal-to-noise ratios (SNR) for multiple noise realizations. He also examined the effects of altering the filter's detection parameter (called beta, β) and the data/transform lengths. It was demonstrated by Go that the Kalman filter did appear to out perform the Hamming window in signal detectability as well as to provide better preservation of the spectral peaks. It is our purpose to take a more in depth look at the Kalman filter used in Reference 4.

1. Filter Parameter (β)

The filter parameter is currently chosen by the user in an experimental fashion. The larger the β the less sensitive the filter is to the dynamics of the spectrum and the smaller the β the more sensitive the filter will behave. There are some obvious trade-offs here. In [Ref. 5], a β=300000 appeared to be the most effective for detecting the spectral peak of signals with FFT output SNR's of 9 dB or greater. The purpose here is to attempt to refine the choose of β for signal detection of signals with FFT output SNR's less than 9 dB.

a. Dynamic Beta

An area of interest is the study of the effects of altering β during the filtering process. If the algorithm is altered so that once a detection occurs the value of β is reduced to make the filter very sensitive (thus preserving the original shape of the periodograms) during the down transition of the spectral peak, then it may be possible to retain the resolvability of two closely spaced spectral components. Similarly, once past a given spectral component, β is reset to its original value and the filtering continues.

b. Modified Kalman Filter

Since we know that spectral components will appear as positive peaks, and usually are of relatively short duration, the Kalman filter algorithm was modified to replace the filter estimate with the original periodogram data while the filter is detecting up transitions in the data.

2. Output Presentation

For very large signal lengths it is cumbersome to look at the periodograms line by line (i.e. transform by transform). It would be useful to somehow stack the periodograms and look at the final outcome of all the transforms in their entirety such as in a 3-Dimensional plot or intensity plot (LOFAR). An additional objective of this research is the development of a display of these stacks of periodograms in a grey-tone image on the Sun workstations.

The graphics package used in this research is NCAR graphics. The LOFAR outputs are created using the subroutine HALFTON which draws a halftone picture from data stored in a rectangular array. These arrays are read in row by row. The subroutine assigns to the largest value in the data the darkest grey-tone (i.e. highest intensity), and conversely assigns the lightest grey-tone to the lowest value in the data. Sixteen intensity levels for the mapping of data can be used in a halftone image. To each intensity corresponds an equal range in the data (i.e. linear mapping). Appendix D is a step-by-step users guide that provides the necessary guidance to produce LOFAR output on a NPS Sun work station. The NPS Sun stations use the UNIX operating system via an Internet Networking System.

II. SPECTRAL ESTIMATION

A. BACKGROUND

Fourier analysis has been an analytical tool for use with analog signal processing for continuous-time signals. Due to the vast improvements made in the computational power of the digital computers, the old school approach is no longer the only method of choice. There are many applications today in analog signal processing where it is preferred to sample the original signal, process it on a digital computer utilizing a discrete-time system, then convert it back to an analog signal if desired. One technique to accomplish this is the discrete Fourier analysis, which is the discrete version of continuous-time Fourier analysis. Fourier-based estimation, also called classical spectral estimation, includes two main nonparametric methods. The first is the periodogram method, which has limited resolution and a variance of the spectral estimate that remains constant independent of increases in data length. Windowing techniques may be used to improve the variance. These techniques are extremely efficient and produce adequate results for many dissimilar types of signals. The second Fourier-based method is the Blackman-Tukey method which utilizes the Fourier transform of an estimate of the autocorrelation function. A problem of the Fourier-based

methods is caused by the windowing of the data during processing. Although windowing minimizes the effects caused by discontinuities of data at the boundaries, a drawback with the use of windows is that leakage in the spectral domain nearly always results. This is where energy in the main lobe of the spectral response leaks into the nearby sidelobes, which can obscure and misrepresent other spectral frequency components that may be present. Using a non-tapered window (i.e., rectangular window) provides the best resolution of any window currently in use although the spectral leakage resulting from using a rectangular window is worse than realized with any other window. [Ref. 6].

B. PERIODOGRAM

There are many procedures used to estimate the spectral content of a sample of a process. The popular choice remains the periodogram primarily due to its ability to economically and effectively be implemented in real time. The periodogram is simply the square of the magnitude of the Fast Fourier Transform (FFT) of a finite duration sequence $x(n)$.

1. Discrete Fourier Transform

Let $x(n)$ be a discrete periodic signal of period N, then the definition for the discrete Fourier series pair is

$$x[n] = \sum_{k=0}^{N-1} a_k e^{jk(2\pi/N)n} \qquad (1)$$

$$a_k = \frac{1}{N} \sum_{n=0}^{N-1} x[n]\, e^{-jk(2\pi/N)n} \quad . \qquad (2)$$

It can be clearly seen in equation (1) that the discrete Fourier series is a finite length of N terms, compared to the infinite number of terms found in the continuous Fourier series. In addition, the largest number of a_k terms present is N, hence $\{a_k\}$ will be a periodic sequence with period N.

There is no discrete Fourier representation for nonperiodic discrete signals. Nonperiodic signals will normally be of infinite length and are represented by the following discrete-time Fourier transform pairs;

$$x[n] = \frac{1}{2\pi} \int_{\Omega_0}^{\Omega_0+2\pi} X(\Omega)\, e^{j\Omega n} d\Omega \qquad (3)$$

$$X(\Omega) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\Omega n} \quad , \qquad (4)$$

where Ω_0 is any real number. The discrete-time Fourier transform of $x[n]$ is $X(\Omega)$ which is a continuous function having a period of 2π.

Since $x[n]$ is normally infinite in length and $X(\Omega)$ is continuous, it is not practical to implement the discrete-time

Fourier transform on the computer. One approach to solving this problem is to make $x[n]$ nonzero only for $0 \le n \le N-1$. This approach lead to the development of a new transform called the discrete Fourier transform (DFT) given by

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\, e^{jk(2\pi/N)n}, \qquad 0 \le n \le N-1 \tag{5}$$

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-jk(2\pi/N)n}, \qquad 0 \le k \le N-1 \quad , \tag{6}$$

where $X[k]$ is the DFT of $x[n]$. [Ref. 6:p.79].

2. Fast Fourier Transform

There are a total of N terms involved in the summation of equation (6) and its computation requires $(N-1)$ complex additions. In addition, each computation in the summation requires one complex multiplication. To compute a N-point DFT of a N-point sequence in a straight forward fashion, $N(N-1)$ complex additions and N^2 multiplications are required. This is a significant computational strain for a computer and much effort was expended to develop ways of making the computation of the DFT faster. The early to mid 1960's saw great progress in this area. The Cooley-Tukey algorithm, the Good-Thomas algorithm, Rader algorithm and chirp z algorithm (all which came to be known as Fast Fourier Transforms), just to name a few, resulted in increased computational speed over the DFT. For a more complete listing of FFT algorithms as well as their

developments see [Ref. 6:pp. 90-115]. All of these algorithms offer features that are particularly useful and efficient when implemented on certain classes of computers.

Regardless of how $X(k)$ is computed, once it is obtained, the periodogram spectral estimate is computed via

$$S_N(k) = |X(k)|^2 = X^*(k)X(k), \quad k = 0,1,\ldots N-1, \tag{7}$$

where $X^*(k)$ is the complex conjugate of $X(k)$.

C. BLACKMAN-TUKEY SPECTRAL ESTIMATION

The Blackman-Tukey spectral estimator was named after R. Blackman and J. Tukey who published their work in 1958. The procedure estimates the autocorrelation function (ACF) as

$$\hat{r}_{xx}^*[k] = \begin{cases} \frac{1}{N}\sum_{n=0}^{N-1-k} x^*[n]\,x[n+k] & k=0,1,\ldots,N-1 \\ \hat{r}_{xx}^*[-k] & k=-(N-1),-(N-2),\ldots,-1 \end{cases} \tag{8}$$

which is Fourier transformed to obtain the PSD estimate. The periodogram is given by

$$\hat{P}_{PER}(f) = \sum_{k=-(N-1)}^{N-1} \hat{r}_{xx} e^{(-j2\pi fk)} \quad . \tag{9}$$

The inferior performance of the ACF estimator is the cause for inferior performance of the Blackman-Tukey estimator. The ACF

estimator is biased regardless of sample size and has an expected value of

$$E[\hat{r}_{xx}[k]] = \frac{N-|k|}{N} r_{xx}[k] \qquad |k| \leq N-1 \quad . \qquad (10)$$

Examination of equation (10) demonstrates that the mean value of the ACF estimator is equal to the true ACF weighted by a Bartlett (triangular) window. An unbiased ACF could be used by replacing the $1/N$ factor in equation (8) with $1/(N-|k|)$. This could result in a negative spectral estimate since a positive semidefinite sequence can not be guaranteed when using an unbiased ACF estimator. This estimation approach was most popular until the development and implementation of the FFT algorithm. [Ref. 7].

D. WINDOWING EFFECTS

Windowing (often referred to as weighting or shading) is an important topic that is integral to all classical estimation methods. We use windows to control the effects of sidelobes associated with these classical spectral estimators. "Tapering functions" is another name often given to data windows. The fundamental purpose of a data window is to lessen the bias in periodogram estimates. [Ref. 4:pp. 136-146].

Processing of finite duration data sets presents unique problems in analyzing the harmonic characteristics of that

data. It is important to pay particular attention to detecting spectral components in the close proximity of stronger spectral components as well as to resolvability of those components. The FFT assumes sequences to be periodic. That is the sampled data being analyzed is one complete period of an infinitely long periodic sequence. Of all possible frequencies in the data set only those that are located at an FFT bin center will be projected as a unique value in the frequency domain. All other frequencies will have non-zero projections over the entire frequency domain. This is termed spectral leakage and is a consequence of data records of finite duration [Refs. 4 and 5].

In nearly all cases of interest, the spectral components present in observed data will be at frequencies other than those located at FFT bin centers. Frequency components not located at a bin center will be non periodic in the observation window. This results in discontinuities at the observation boundaries, which in turn results in leakage over the entire range of the FFT.

Consider a finite data record that is a portion of an infinite sequence that has been multiplied by some window. Consider an observed data sequence $x_0[n]$ of N points that is the product of a rectangular window, $rec(n)$, of unit amplitude and an infinite-duration sequence, $x(n)$, where

$$rec(n) = \begin{cases} 1 & n=0,1,2,\ldots,(N-1) \\ 0 & otherwise \end{cases} \quad (11)$$

and the observed data sequence is

$$x_0[n] = x[n] \cdot rec[n] \quad . \quad (12)$$

The assumption made here is that all unobserved samples are essentially zero. Therefore, data that is processed "as is" is data that has been rectangularly windowed. Window functions other than the rectangular window function are called weighting functions that reduce the effects of boundary discontinuities. This smoothing of the observed data at the boundaries is one goal of windowing.

Multiplication of the time series in equation (12) results in the convolution of the transforms of *x(n)* and *rec(n)* in the frequency domain. This results in a broadening or smearing of the power of the spectral components into adjacent frequency bins, provided the spectral component is not located at a bin center. If non-rectangular windows are used smearing will occur regardless of spectral component location. The narrowest spectral response of a windowed sequence can be no less than that determined by the mainlobe of the transform of the window, independent of the data [Ref. 4:p.137]. These mainlobe widths are different for each window function. For example, the main lobe width (half-power bandwidth, which is 3 dB down from the peak response) for a rectangular window is

about the reciprocal of the observation interval (NT, where T is the time interval between samples). The leakage mentioned earlier results in sidelobes of the main spectral components. These sidelobes may bias the amplitudes of adjacent frequency responses and may completely mask the presence of weaker signals, thus preventing their detection.

Another goal of tapered windows is to achieve better sidelobe levels than those of the rectangular window. By decreasing the sidelobe levels the bias can be reduced. The drawback here is that sidelobe depression can only be accomplished by broadening the windows mainlobe frequency response. This in turn reduces the spectral resolution. Table I presents performance characteristics of different window functions. The trade-off between sidelobe depression and mainlobe resolution must be considered when determining the optimum window of choice.

Table I. Window Performance Characteristics [Ref. 4: p. 143].

WINDOW NAMES	HIGHEST SIDELOBE LEVEL	1/2-POWER BW (DFT BINS)
Rectangle	-13.3 dB	0.89
Triangle	-26.5 dB	1.28
Hann	-31.5 dB	1.44
Hamming	-43.0 dB	1.30

E. ZERO PADDING

There are two primary purposes for zero padding a data set. First, to allow for transform value interpolation between the original N transform values. To illustrate, let us consider a data set $x(n)$ of a finite length N. Next, zero pad that data set with N zeroes such that

$$y(n) = \begin{cases} x(n) & n=0,1,2,\ldots,(N-1) \\ 0 & n=N,(N+1),\ldots,(2N-1). \end{cases} \quad (13)$$

From equation (6) the DFT of the $2N$ point data sequence $y(n)$ is

$$\begin{aligned} Y[k] &= \sum_{n=0}^{2N-1} y[n]\, e^{-jk(2\pi/2N)n} \\ &= \sum_{n=0}^{N-1} x[n]\, e^{-jk(2\pi/2N)n}, \qquad k=0,1,2,\ldots,(2N-1). \end{aligned} \quad (14)$$

Comparing the results of equations (6) and (14) note that

$$Y[k] = \begin{cases} X[k/2] & k=0,2,4,..,(2N-2) \\ \sum_{n=0}^{N-1} x[n]\, e^{-jk(2\pi/2N)n} & k=1,3,5,..,(2N-1) \end{cases} \quad (15)$$

and that the $2N$-point DFT of $y(n)$ is identical to the N-point DFT of $x(n)$ at even index values and therefore the odd values of k represents the interpolated DFT values between the original N-point DFT [Ref 4:pp. 43-44]. The second purpose of

zero padding is to augment a data set to ensure it is of a length that is a power of two to allow the use of an FFT.

To demonstrate the effects of zero padding consider a 16-point rectangular window. The DFT of a rectangular window will result in a digital sinc function of the form

$$D_n(f) = Te^{(-j2\pi ft[N-1])}\left[\frac{\sin(\pi fTN)}{\sin(\pi fT)}\right] \qquad (16)$$

The digital sinc function (D_N) possesses sidelobes. Nevertheless, when we calculate and plot the transform we notice only a central spike at the origin with no sidelobes evident (Figure 1). The reason that no sidelobes are visible is because the FFT of the non-zero-padded rectangular window series examines D_N at its zero crossings and therefore, the structure of the sidelobes remains hidden. The zero crossings of D_N occur at FFT bin centers and by zero padding one can calculate interpolated values between bin centers which allows the presentation of the sidelobe structure. Figure 2 represents the magnitude of the FFT of a 16-point rectangular window zero-padded to 32-points. It is important to understand that zero-padding a data sequence prior to the DFT will not improve the resolution of the periodogram. This is the same principle that applies to more typical spectral estimation problems.

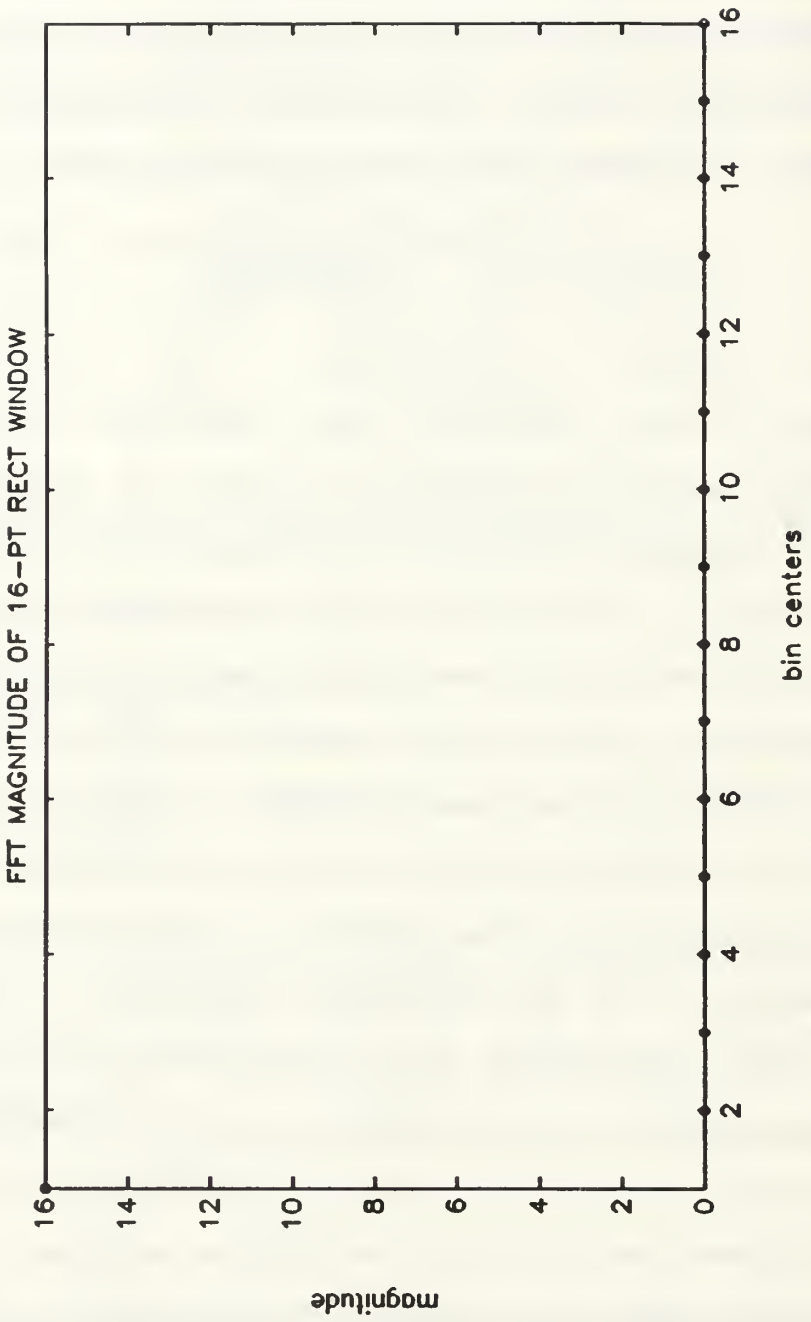

Figure 1. Magnitude of the FFT output of a 16-point Rectangular Window

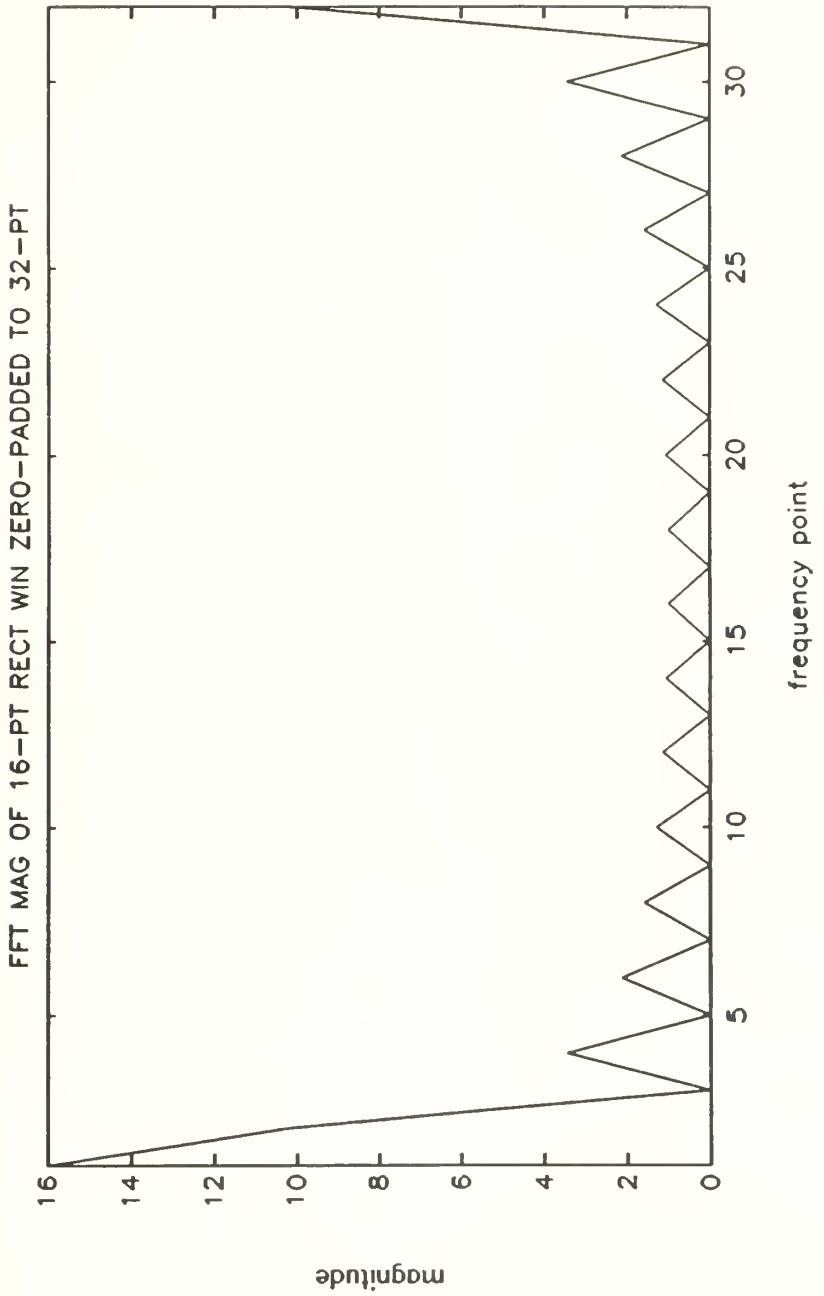

Figure 2. Magnitude of the FFT output of a 16-point Rectangular Window zero-padded to 32-point

To better illustrate the effects of zero-padding consider a sinusoid of unit amplitude intermixed in Gaussian white noise. The noise has a variance of 0.0005 which corresponds to SNR of 30 dB defined by

$$SNR = 10 \cdot \log_{10}\left[\frac{A^2}{2\sigma^2}\right], \tag{17}$$

where A = sinusoidal amplitude and σ^2 is the variance of the Gaussian noise. In this illustration the data record length (N) is 32 and sampling frequency (f_s) is 32 Hz. The sinusoidal frequency used is 10 Hz. The bin centers of the FFT will fall at integer multiples of f_s/N. In this example, f_s/N = 32/32 = 1 Hz per bin. The sinusoidal frequency of 10 Hz is located exactly at a FFT bin center. The spectral peak is clearly displayed (Figure 3) since the frequency is at bin center. Also note virtually no evidence of the presence of sidelobes since the digital sinc function was interrogated at the zero crossings. Figure 4 is the results of the same signal described above with the exception of being zero padded to 64-points. Note that the sidelobes of D_N are now visible because of the zero-padding.

Now let us examine the effect of a sinusoidal frequency not located at bin center. Let the sinusoidal frequency be 10.7 Hz. Figure 5 is the result of a non-zero-padded 32-point sequence. The sidelobes are present even though the sequence

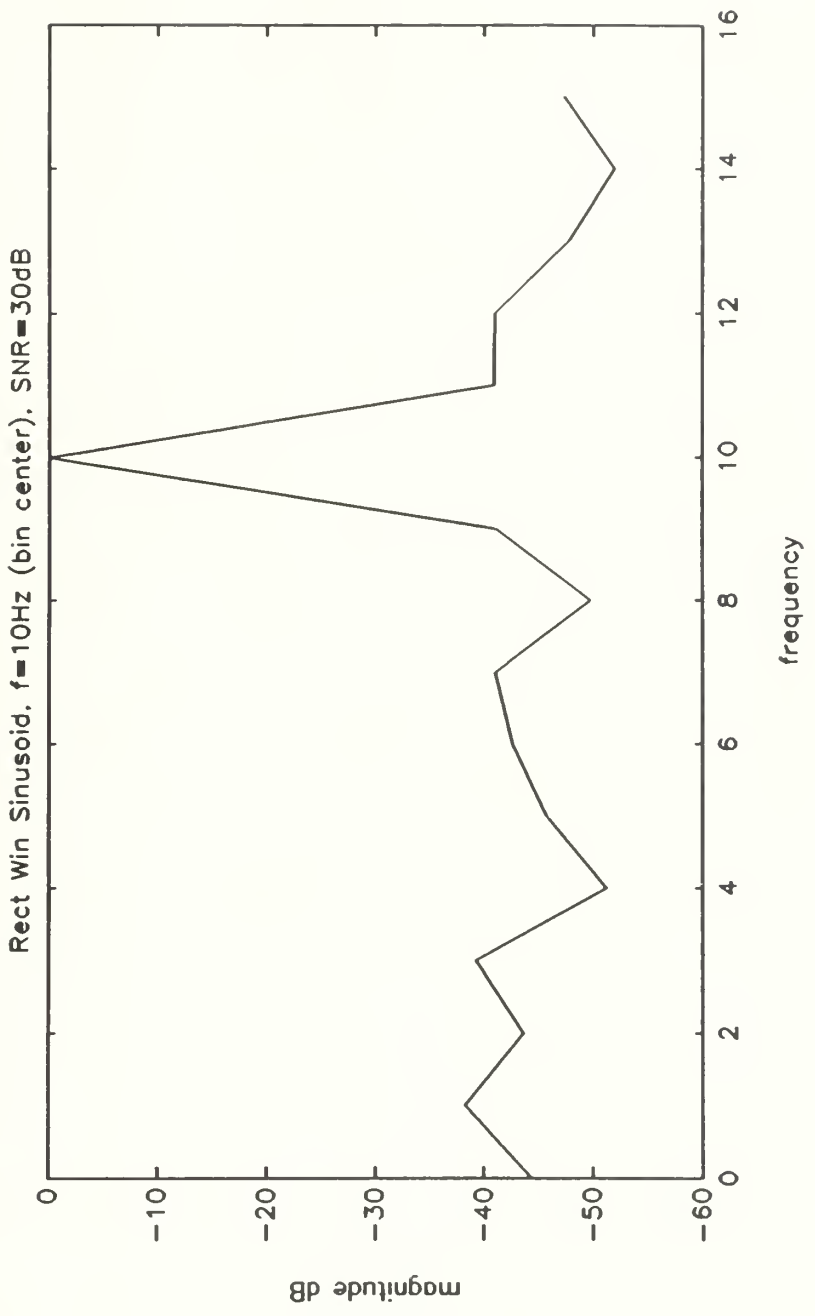

Figure 3. Periodogram of sinusoid in Gaussian noise, f=10 Hz (bin center), 32-point sample, non-zero-padded, f_s=32 Hz

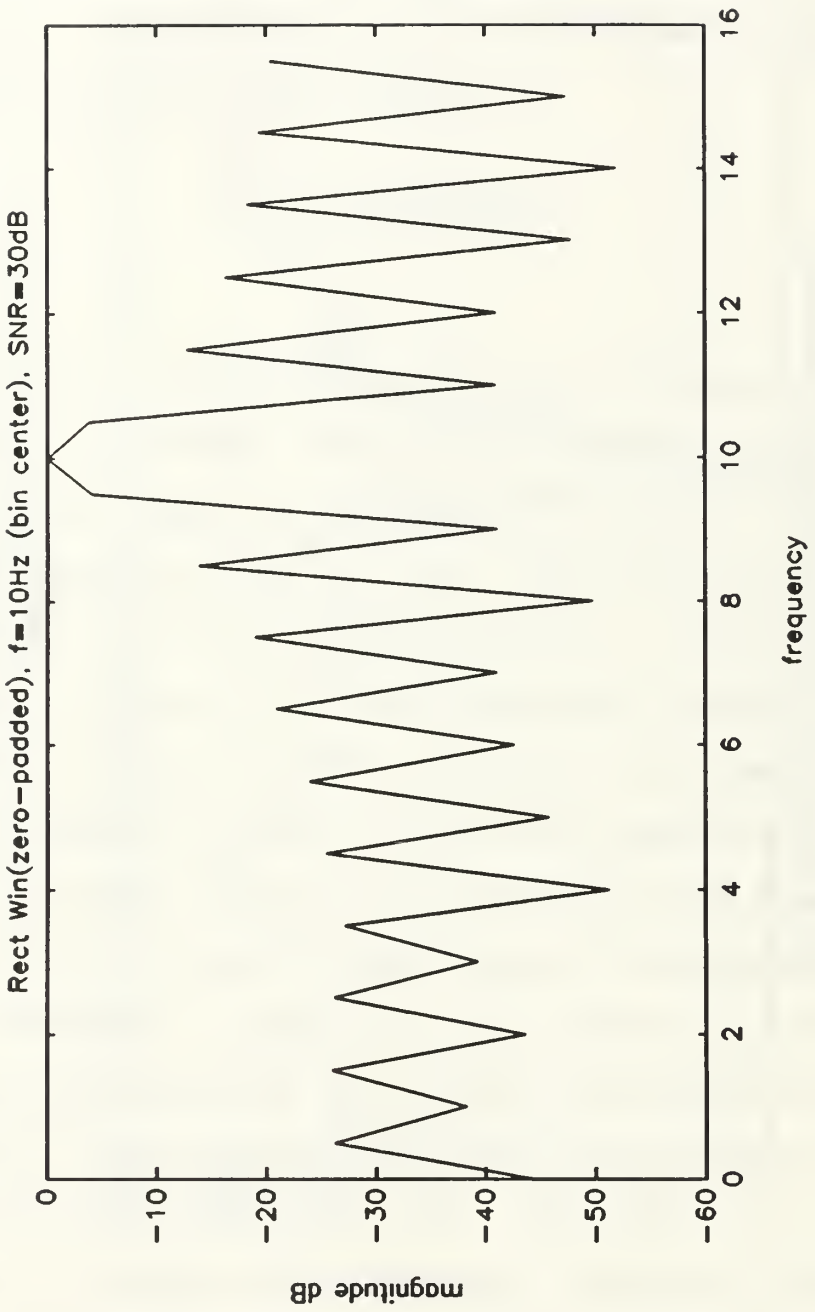

Figure 4. Periodogram of sinusoid in Gaussian noise, f=10 Hz (bin center), 32-pt sample zero-padded to 64-pt, f_s=32 Hz

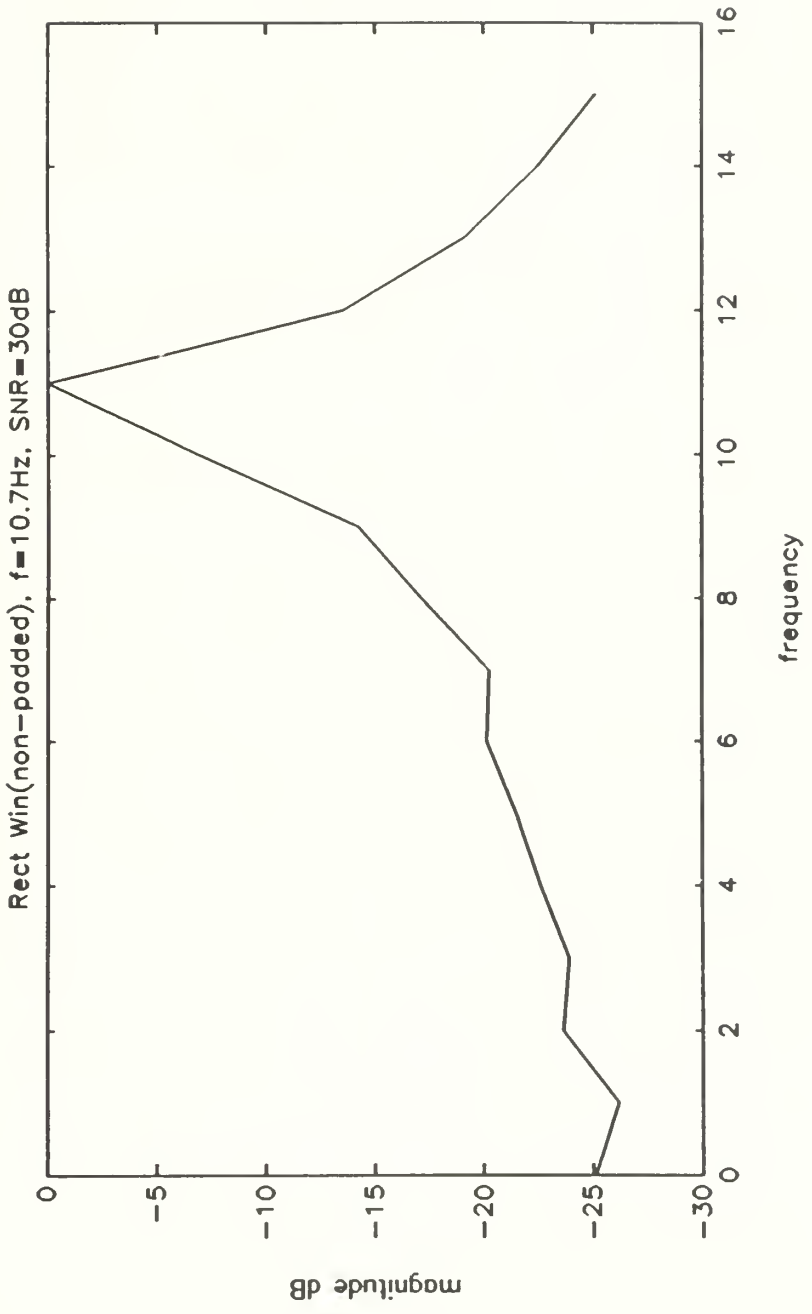

Figure 5. Periodogram of sinusoid in Gaussian noise, f=10.7 Hz (not at bin center), 32-pt sample non-zero-padded, f_s=32 Hz

was not zero-padded. This is because the sidelobes of D_N are now visible since the FFT is interrogating D_N at locations other than its zero crossings. Spectral leakage is also evident, which has smeared signal energy into nearby frequency bins. Notice a broader, less-pronounced mainlobe results.

F. STATISTICAL PROPERTIES OF THE PERIODOGRAM

The distribution of a white Gaussian noise data set $x(n)$ is given by:

$$x(n) \sim N(\mu, \sigma_x^2) \quad . \tag{18}$$

Consider the case where the mean of $x(n)$ is zero (i.e. μ=0) and the sample is of size N. The DFT of $x(n)$, defined as $X(k)$, will be comprised of real and imaginary parts, labeled $A(k)$ and $B(k)$ respectively, that are orthogonal linear combinations of $x(n)$. For simplicity let $X(k)$ be normalized by 1/SQRT(N). Therefore, both $A(k)$ and $B(k)$ are Gaussian random variables with the distribution $N(0,\sigma_x^2)$ and are mutually uncorrelated. The sum of the squares of the real and imaginary parts of $X(k)$ is the periodogram (Per_x) of $x(n)$ and is given by

$$Per_x(k) = A^2(k) + B^2(k) \quad . \tag{19}$$

The probability density function of $Per_x(k)$ is chi-squared with 2 degrees of freedom. The mean and variance of $Per_x(k)$ are

$$E\left[Per_x(k)\right] = 2\sigma_x^2 \qquad for\ all\ k \tag{20}$$

$$Var\ [Per_x(k)] = \begin{cases} 4\sigma_x^2; & k \neq 0, \frac{N}{2} \\ 8\sigma_x^2; & k = 0, \frac{N}{2} \end{cases} . \tag{21}$$

For a derivation of equations (20) and (21) see [Ref. 8].

III. SPECTRAL ESTIMATION USING THE KALMAN FILTER

A. BACKGROUND

As described in Chapter II the problem with the use of FFT-based spectral estimation is the trade-offs associated with sidelobe suppression versus spectral resolution. The application of tapered windows to a time series data set for the purpose of suppressing the sidelobes results in the broadening of the spectral mainlobe. Each tapered window possesses different performance characteristics as presented in Table I. It was demonstrated by Go [Ref. 5] that the application of the Kalman filter to the periodogram of an unwindowed spectral record for minimizing the variance with minimal degradation of the spectral resolution showed promise. Figure 6 allows the comparison of the resolution maintained by the Kalman filter relative to the mainlobe broadening seen in the Hamming window. The conclusions reached by Go were:

1. The Kalman filter can enhance the spectral peaks of the periodogram of an unwindowed time series.
2. The resolution for the multi-spectral peaks of the periodogram is largely preserved by the Kalman filter.
3. Reliable signal detection was achieved for signals with SNR's (after processing) of 12 dB for β in the range between 100,000 and 700,000.
4. Signal detection was possible for SNR's (after processing) down to 6 dB in a small percentage of noise realizations.

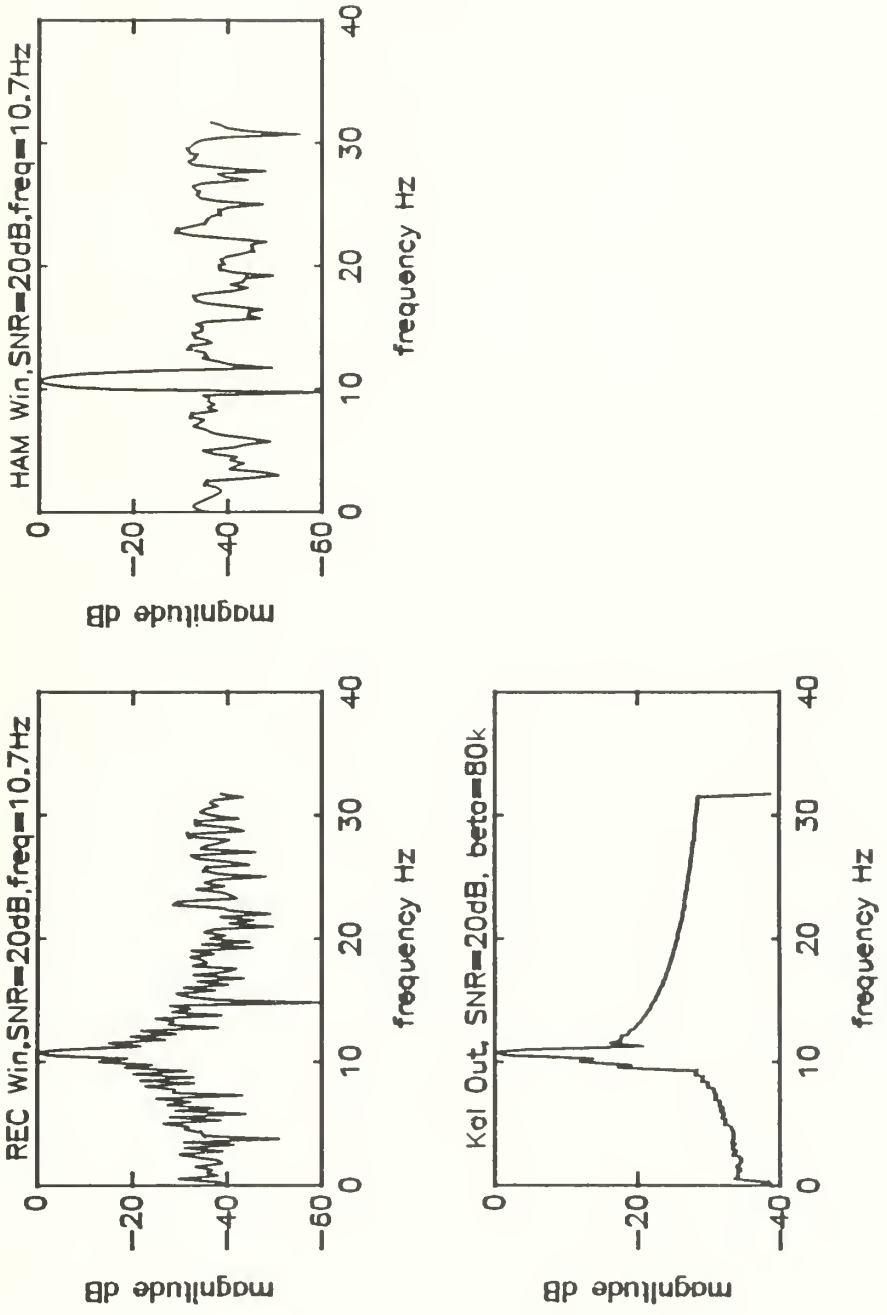

Figure 6. Periodogram of f=10.7 Hz, input SNR=20 dB, and filter using Hamming window and Kalman filter β=80K

Kalman filtering is an estimation or prediction procedure which was introduced by R.E. Kalman and R.S. Bucy in the early 1960's. The basic idea of a Kalman filter is to recursively update the estimate of the state of a system by comparing these estimates to the system measurements. A new estimate follows each measurement. The basic ingredients of the Kalman filter are a stochastic differential system (called the system model) and a set of observations (called the measurement model) which allow for the computation of the state vector estimates. The measurement and the state estimate need not be of the same dimensionality. [Ref. 9].

The Kalman filter algorithm demonstrated by Go was written by Dr. Roberto Cristi at the Naval Postgraduate School, Monterey California in 1988. The algorithm was developed to detect piecewise constant segments of time series data corrupted by noise. If data under examination consists of piecewise constant segments, then there are two possibilities at each new observation. They are;

1. the current observation is a continuation of the previous piecewise constant segment of data or
2. the current observation is the first element of a new segment of data.

The algorithm uses a filter parameter β to quantify the likelihood of a transition from one piecewise constant segment to another. If no transition is detected then the current observation is filtered using a Kalman filter updated with the current Kalman gain. If a transition is detected then the

current observation is filtered using a reinitialized Kalman filter. The sensitivity of the filter to detect transitions between piecewise constant segments is a function of the parameter β. If the value of β is "too small", then the filter will reinitialize too often resulting in less smoothing of the data. If the value of β is "too large", then over-smoothing will occur and transition points will not be detected. For a detailed description of the development of this algorithm see [Ref. 5: ch. 3]. To demonstrate the effects of different values of β consider a sinusoid corrupted by Gaussian noise. Figure 7 shows the result of filtering a periodogram of an unwindowed sinusoid (freq=10.7 Hz) time series with a sample size of 128-pts zero padded to 256-pts sampled with f_s (sampling frequency)=64 Hz and an input SNR=-9 dB. Values for β of 8k, 80k and 800k (where $k=10^3$) where used to filter this data. Note the added detail in the plot of the filter output for β=8k versus the over smoothing where β=800k.

B. EXPERIMENTAL DATA

The data used in this research was computer generated. It is sinusoidal data in additive Gaussian white noise of different variances. Every data set had at least one sinusoid of unit amplitude with a preselected SNR. The noise variance (σ^2) for that data set was calculated using equation (17). For data sets with more than one frequency component, the

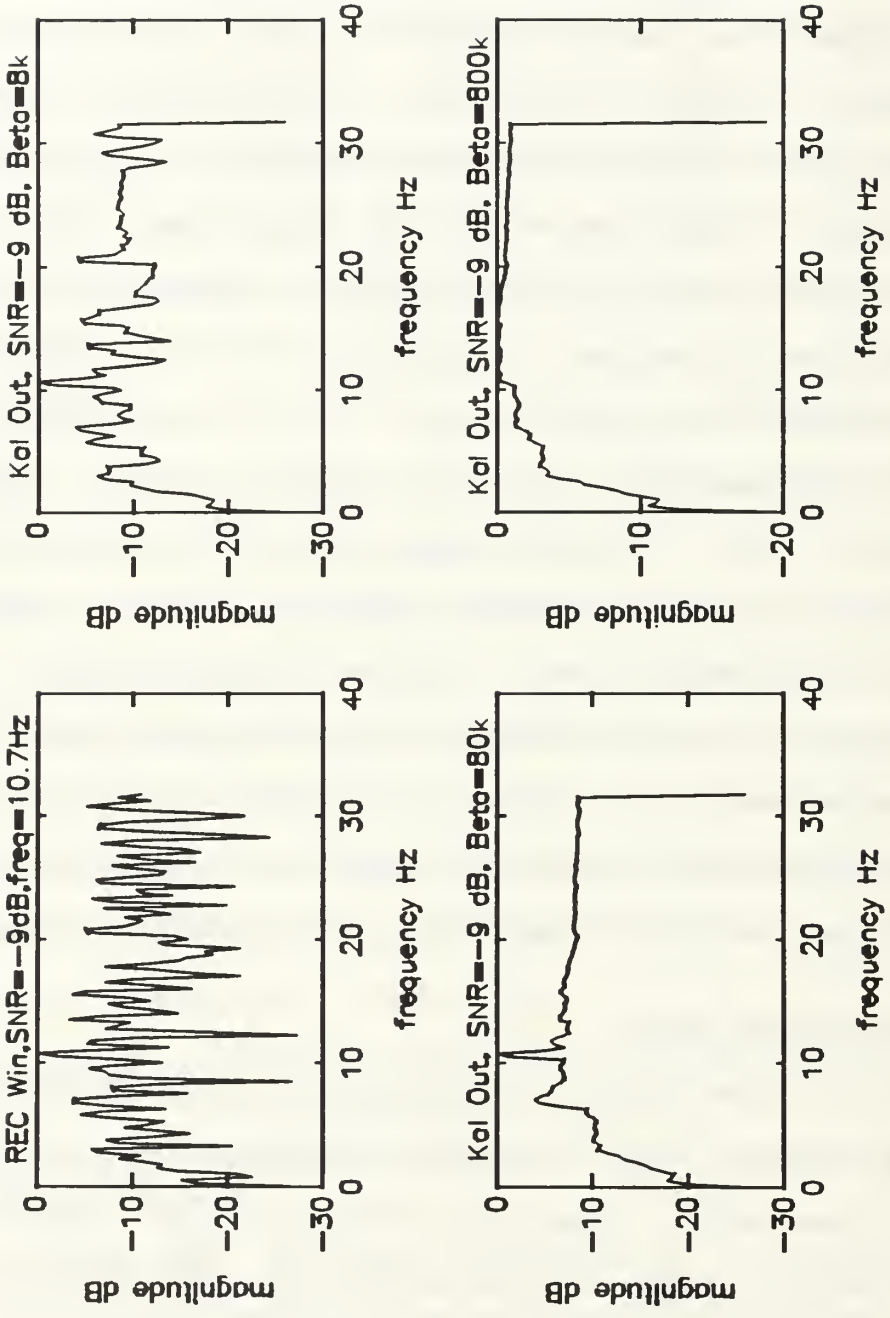

Figure 7. Periodogram of 10.7 Hz sinusoid of -9 dB input SNR filtered for β values of 8k, 80k, 800k respectively.

amplitude of each component was the determining factor in achieving that component's desired SNR.

Transform lengths of 128-pts zero-padded to 256-pts were used exclusively in this research. Zero-padding does not improve FFT processing gain, which for best case can be approximated by

$$[\log_2(\textit{data record length}) - 1] \times 3 \ \ dB \qquad (22)$$

provided the data is stationary over the data length. The processing gain for an 128-pt FFT is about 18 dB, therefore the realized SNR on the output side of the FFT is

$$\begin{aligned} SNR_{out} &= SNR_{in} + PG \\ &\approx SNR_{in} + 18 \end{aligned} \qquad (23)$$

where SNR_{out} is the SNR on the output side of the FFT and SNR_{in} is the SNR of the input signal. Data used in this study will be of this form unless otherwise specified.

C. KALMAN FILTER MODIFICATIONS

1. Dynamic Beta

It was shown [Ref. 5] that the Kalman filter algorithm can be successfully adapted for smoothing spectral data. The filter parameter β is the controlling factor that determines how much the noise portions of the periodogram are smoothed and how well the transition points are detected. The result

for an appropriately chosen β will be a smoothed periodogram with narrow mainlobes that are essentially unaltered from the original periodogram. A common trait of the original Kalman filter is a tapering (i.e., roll-off) effect on the higher frequency side of the mainlobe. This is a result of the Kalman filter failing to reinitialize after the peak and on the down-transition of the periodogram. A smaller valued β would be more sensitive and would more closely follow the down-transition of the original periodogram. Unfortunately, as shown in Figure 7, small values of β provide less smoothing and little improvement over the original periodogram.

This is the motivation for allowing β to be dynamic during the filtering process. An optimum value of β is used during the initial filtering process until a transition is detected. Once a transition is detected the value of β is decreased to insure maximum sensitivity (β=1 is used in this study), which minimizes the tapering effect on the down-transition side of the spectral component.

Consider a sinusoid where f=10.7 Hz and SNR_{in}=-9 dB. The periodogram of the unwindowed data record is filtered using the original Kalman filter algorithm with β=80k. The same periodogram is again filtered using the Kalman filter with the dynamic β (i.e., β=80k or 1). Figure 8 shows the result of that type of processing. Notice the tapered response on the higher frequency side of the spectral peak in

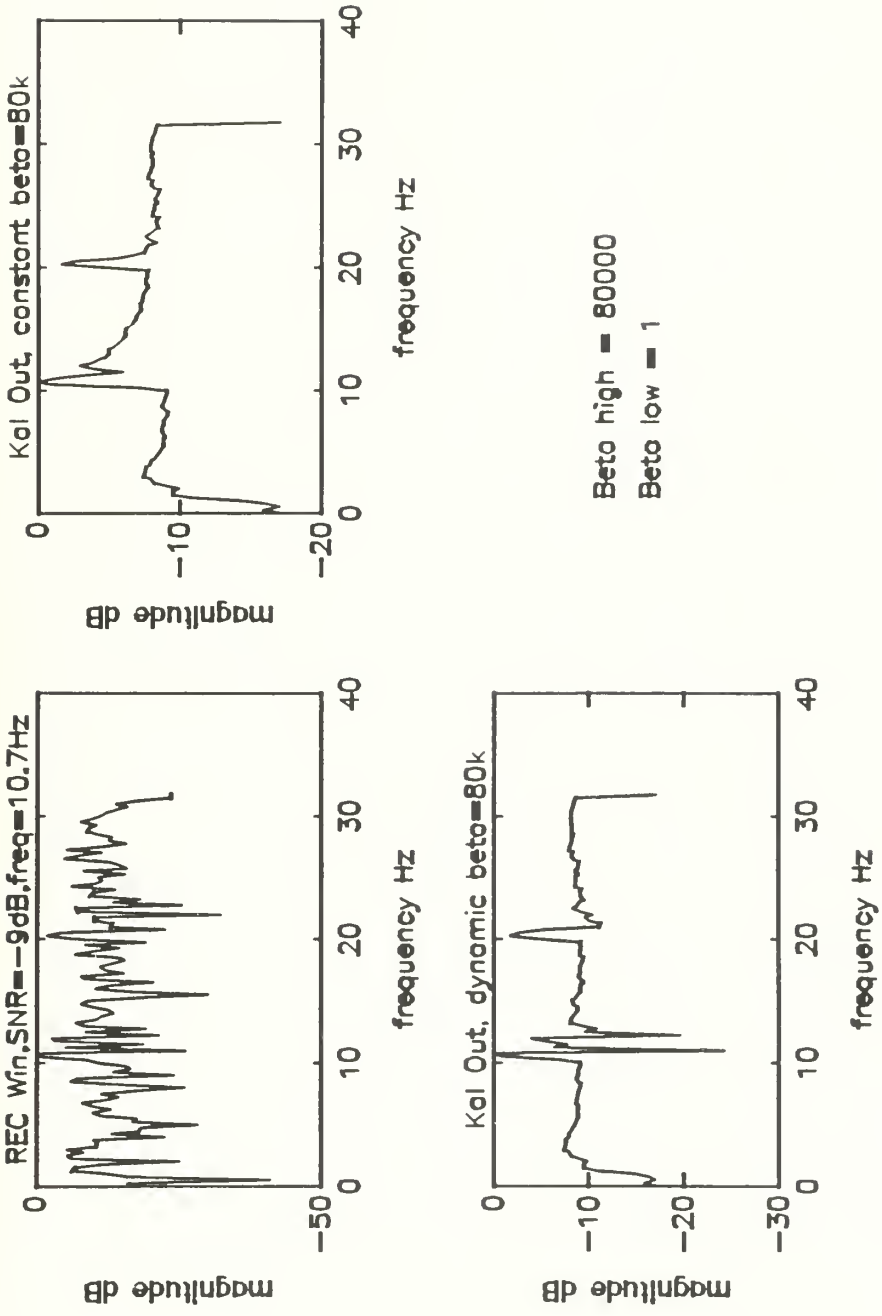

Figure 8. Periodogram of 10.7 Hz sinusoid of SNR_{in}=-9 dB filtered with constant β=80k and a dynamic β=80k

the case for constant β filtering. The shape of the unwindowed periodogram on the high frequency side of the mainlobe is clearly more preserved in the dynamic β case. Another important observation is that there is no change in the dB level of the noise floor for either filter output.

It is not uncommon for the dynamic β filter to provide similar results as when using the constant β filter for some noise realizations. While β is at the "very" sensitive setting and the data does not exceed a given transition threshold then the algorithm will redefine β to the original filtering value. This redefining of β may occur in the vicinity of the spectral peak. This can result in filtering of the down side of a spectral peak with the same value of β used in the constant β filter.

Although there are instances when the dynamic β will not improve filter performance, it was exceedingly rare (less than 10 out of 256 trials) for the dynamic filter to provide less satisfactory results than the constant β filter. The LOFAR output is the presentation of 128 single step traces. Even if only a few of the 128 traces are improved by the dynamic filter process, then those improvements could lead to enhancement of signal detectability and resolution.

2. Data Substitution

The second modification made to the Kalman filter algorithm was driven by similar motivation as for the dynamic

β filter discussed earlier. It is clear that the Kalman filter can smooth spectral data while leaving the mainlobe of the periodogram virtually unaltered for appropriately chosen β and SNR_{out} of 9 dB and larger. The second modification was made in the attempt to improve upon the filters ability to preserve resolution of the original spectral data. In the frequency domain the spectral component of a sinusoid will span only a small number of FFT bins as compared to the total number of bins/frequencies represented in the FFT output. For an appropriately chosen β the total number of times the Kalman filter detects transitions from one piecewise smooth segment to another is few compared to the number of times the filter detects no transitions.

Let $Y(k)$ be the periodogram of an unwindowed time series and let $X(k)$ be the output of the Kalman filter (for either the constant β or dynamic β algorithm) computed from the filtering of $Y(k)$. The Kalman filter was modified to produce a new output ($X_M(k)$)

$$X_M(k) = \begin{cases} X(k) & \textit{no transition detected} \\ Y(k) & \textit{up-transition detected} . \end{cases} \qquad (24)$$

This modification can be implemented in conjunction with either the dynamic or constant β filters. The notation "modified" Kalman will be used to describe this modification throughout this study.

Figure 9 shows the Kalman output of a dynamic β process versus the output from the dynamic β algorithm modified as described above in equation (24). Notice there is not an obvious improvement in resolution of the spectral peak over the dynamic β only case (upper right Figure 9), but there is a measurable lowering of the noise floor realized for the dynamic β filter modified to substitute raw data in for the estimate during detected transitions. The noise realization filtered in Figure 9 displays about a 1.5 dB improvement in raising the spectral peak height above the noise floor. This is a consistent result observed throughout this study.

D. KALMAN VERSUS HAMMING

Results presented by Go [Ref. 4] indicated that the Kalman filter appeared to outperform the Hamming window in signal detection at SNR_{out} of 9 dB and below for β=300k. The relatively small number of noise realizations looked at in reference 4 were insufficient to estimate just how much better Kalman was over the Hamming window in signal detection. Here the outputs of the Hamming window and Kalman filters (for β=80k,135k and 300k) of both the dynamic β and the "modified" Kalman filter using dynamic β were examined for 128 noise realizations. This was done twice. Once for SNR_{in}=-9 dB and once for SNR_{in}=-12 dB. For both cases the sinusoidal frequency was f=20.0625 Hz. The output for each noise realization was examined to determine a hit or miss for signal detection for

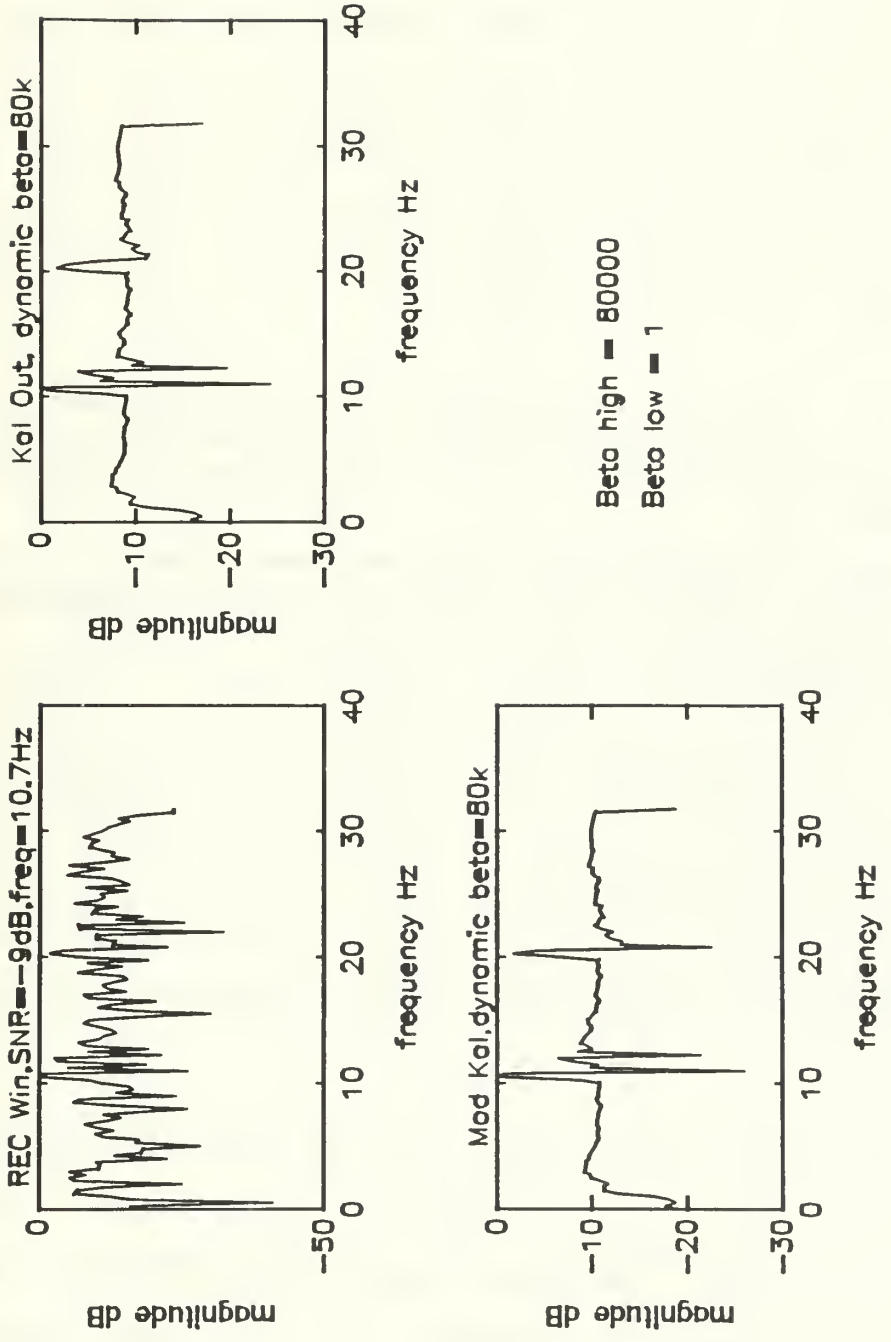

Figure 9. A comparison of the performance of Kalman filtering (dynamic β vs modified dynamic β) of an periodogram of an unwindowed time series of f=10.7 Hz, SNR_{in}=-9 dB

each filter process. An example of one noise realization for SNR_{in}=-9 dB for the Hamming window (upper right), Kalman with dynamic β=80k (lower left), and Kalman "modified" filter with dynamic β=80k (lower right) is provide with Figure 10. If the output displayed a peak at the true spectral location then a detection was recorded for that filter process and that noise realization. The totals were tabulated and are presented in Table II.

Table II. Detection comparisons for Kalman vs Hamming for 128 noise realizations at SNR_{in}=-9 dB and for 128 noise realiza-tions at SNR_{in}=-12 dB. The number of misses are provided.

Filter Process	SNR_{in}=-9 dB	SNR_{in}=-12 dB
Hamming Window	31	59
Dyn β = 80k	19	52
Mod Kal, Dyn β=80k	19	49
Dyn β = 135k	24	59
Mod Kal, Dyn β=135k	24	52
Dyn β = 300k	36	70
Mod Kal, Dyn β=300k	36	67

A statistic that was not collected was the number of false detection (i.e., false alarms) made by each different filtering process. Since the spectral location of the false alarms are random and independent of the noise realization, and each trace is one of 128 that will be displayed in a LOFAR output, the false detections should have negligible impact.

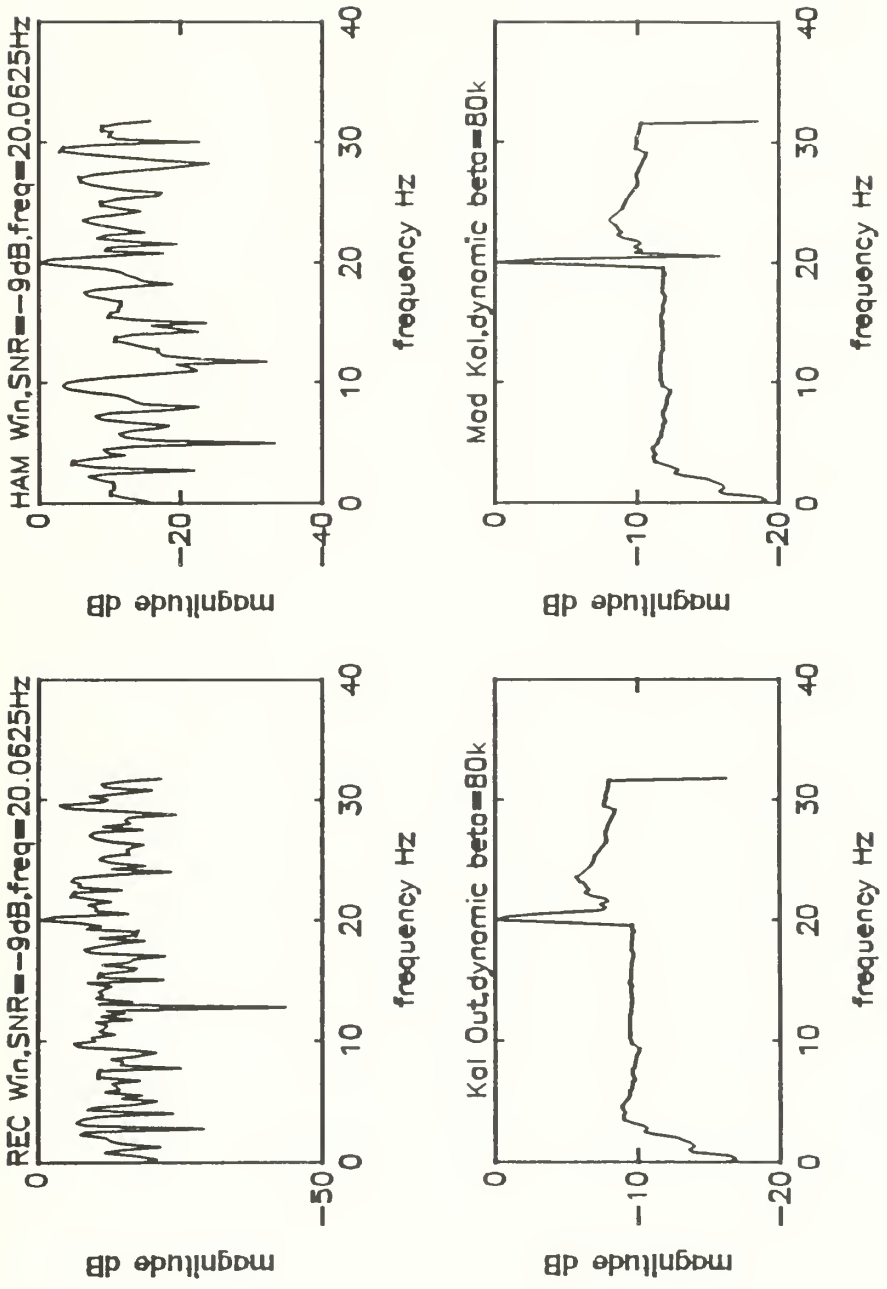

Figure 10. Periodogram (f=20.0625 Hz, SNR_{in}=-9 dB) filtered using Hamming window, Kalman Filter Dyn β=80k and Modified Kalman Filter Dyn β=80k

E. LOFAR OUTPUT

As mentioned earlier, for very long signal segments it can be cumbersome and time consuming to look at the periodograms or filtered periodograms trace by trace. Also at SNR_{out} of 6 dB or less it is sometimes difficult to distinguish signal from noise for many of the individual traces. Another problem of examining spectral data trace by trace is the difficulty of detecting the dynamics that may be present in some signals.

A solution to these problems is to display all (or at least a substantial number) of the periodograms simultaneously. This display needs to be relatively fast and compatible with the SUN workstations. The procedure used is to stack the periodograms into a matrix that is 128 by 128 in size. The largest numerical value in the matrix is assigned the darkest grey-tone and conversely, the lightest grey-tone is assigned to the lowest value. Sixteen intensity levels are used for the mapping of the remaining data. To each intensity level is a corresponding equal range in the spectral density.

The graphics package used to create this image is NCAR. NCAR Graphics is a collection of FORTRAN 77 programs and subroutines that can be used to generate and plot computer graphics suitable for the display of scientific data. NCAR Graphics conforms to the Graphical Kernel System (GKS) standard, Level OA (zero A).

The LOFAR outputs used in this study are created using the subroutine HALFTON which draws a halftone picture from data stored in square or rectangular arrays. A time domain (sinusoidal) signal corrupted by additive Gaussian white noise is generated. The signal length is 16384 points. The signal is processed in segments using 128 non-overlapping contiguous samples, zero-padding to 256 points. Each segment provides one periodogram which is stacked in array form. The resulting array is a square matrix 128 by 128, where the rows are the periodograms of the individual segments and the columns represent frequency. The matrix is read into HALFTON and is displayed as a LOFAR output. Figure 11 is an example of a LOFAR output of rectangular windowed data with f=10.7 Hz, SNR_{in}=-12 dB and f_s=64 Hz. The horizontal axis is the frequency axis which goes from 0 to 32 ($f_s/2$) Hz while the vertical axis represents time running from the bottom to the top. The 10.7 Hz line is quite obvious even though the SNR_{out} is only 6 dB. This dramatic increase in detectability over the individual periodograms is due to the additional processing gain obtained by the visual display.

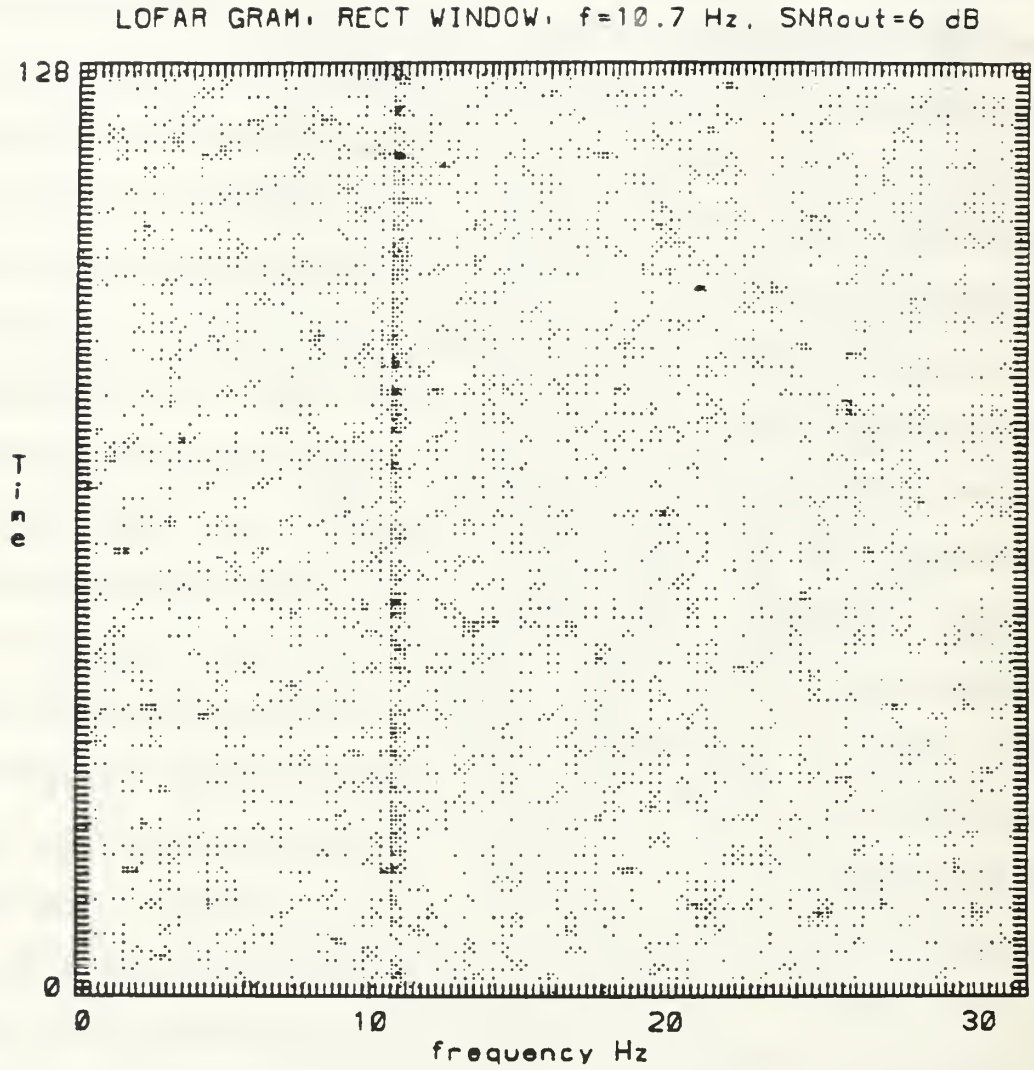

Figure 11. LOFAR output of rectangular windowed signal; f=10.7 Hz, SNR_{in}=-12 dB and f_s=64 Hz

IV. EXPERIMENTAL RESULTS

A. EFFECTS OF FILTER MODIFICATIONS

1. Dynamic versus Constant β

Allowing β to be dynamic during the filtering process often results in a more accurate representation of the spectral shape on the higher frequency side of the mainlobe than achieved by the constant β filter. Figure 12 thru Figure 21 are 10 different noise realizations which compare the performance of a constant β filter with a dynamic β filter. The spectral data has two frequency components. One is f1 at 10.7 Hz at an SNR_{in} of -9 dB and the other one is f2 equal to 11.7 Hz at an SNR_{in} of -12 dB. A 1 Hz separation is equivalent to 4 bin separation for a sample size of N=128 at a sampling frequency of 64 Hz. The dynamic β filter provided more accurate representation of the original periodogram in 4 of the 10 noise realizations without a single case of inferior performance.

Figure 22 shows two LOFAR outputs allowing the comparison of the outputs for constant and dynamic β. In this representation it is not clear that there is any distinguishable difference between the two filtering outputs.

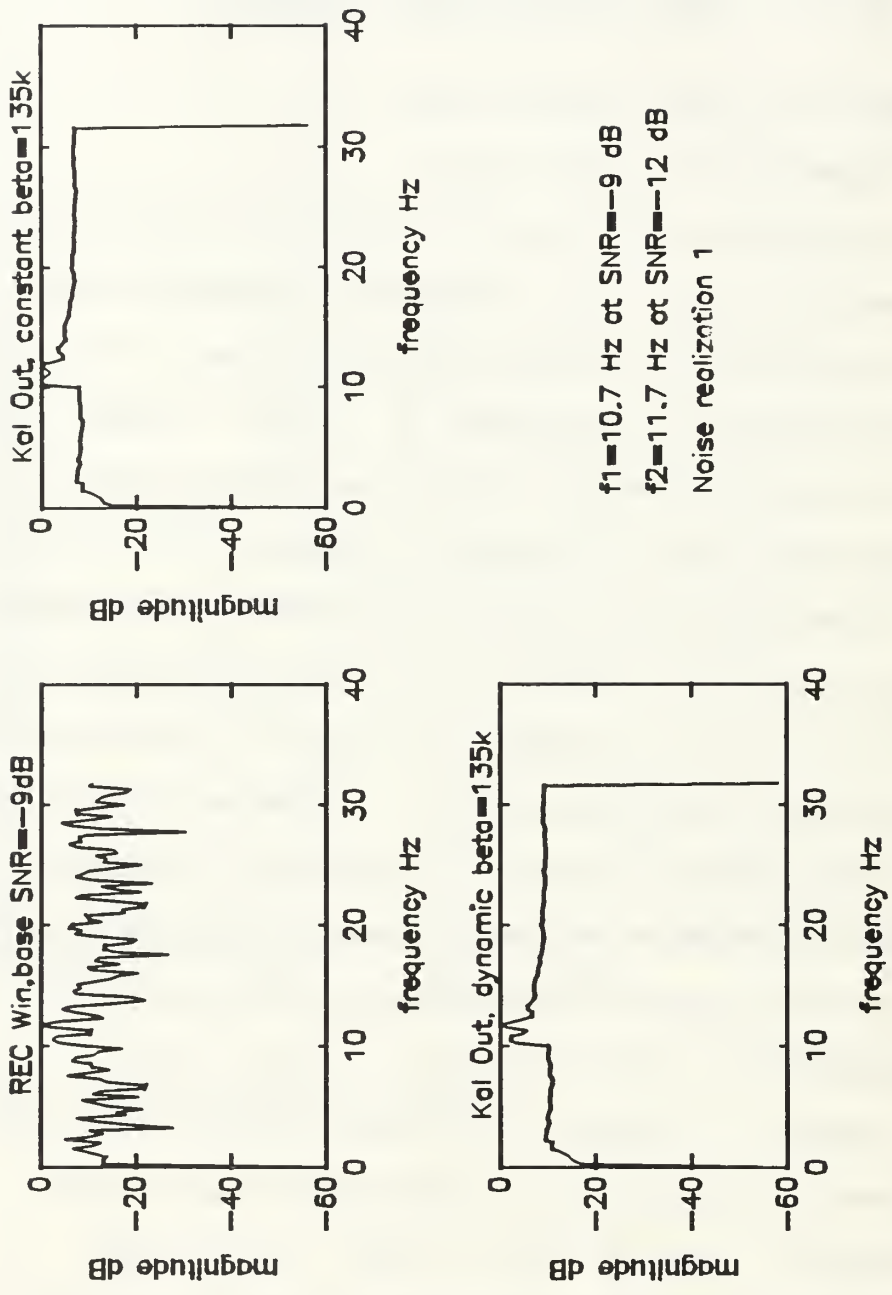

Figure 12. Periodogram, Noise Realization 1 (f1=10.7 Hz at SNR_{in}=-9 dB and f2=11.7 Hz at SNR_{in}=-12 dB). Kalman filter using a constant β and a dynamic β.

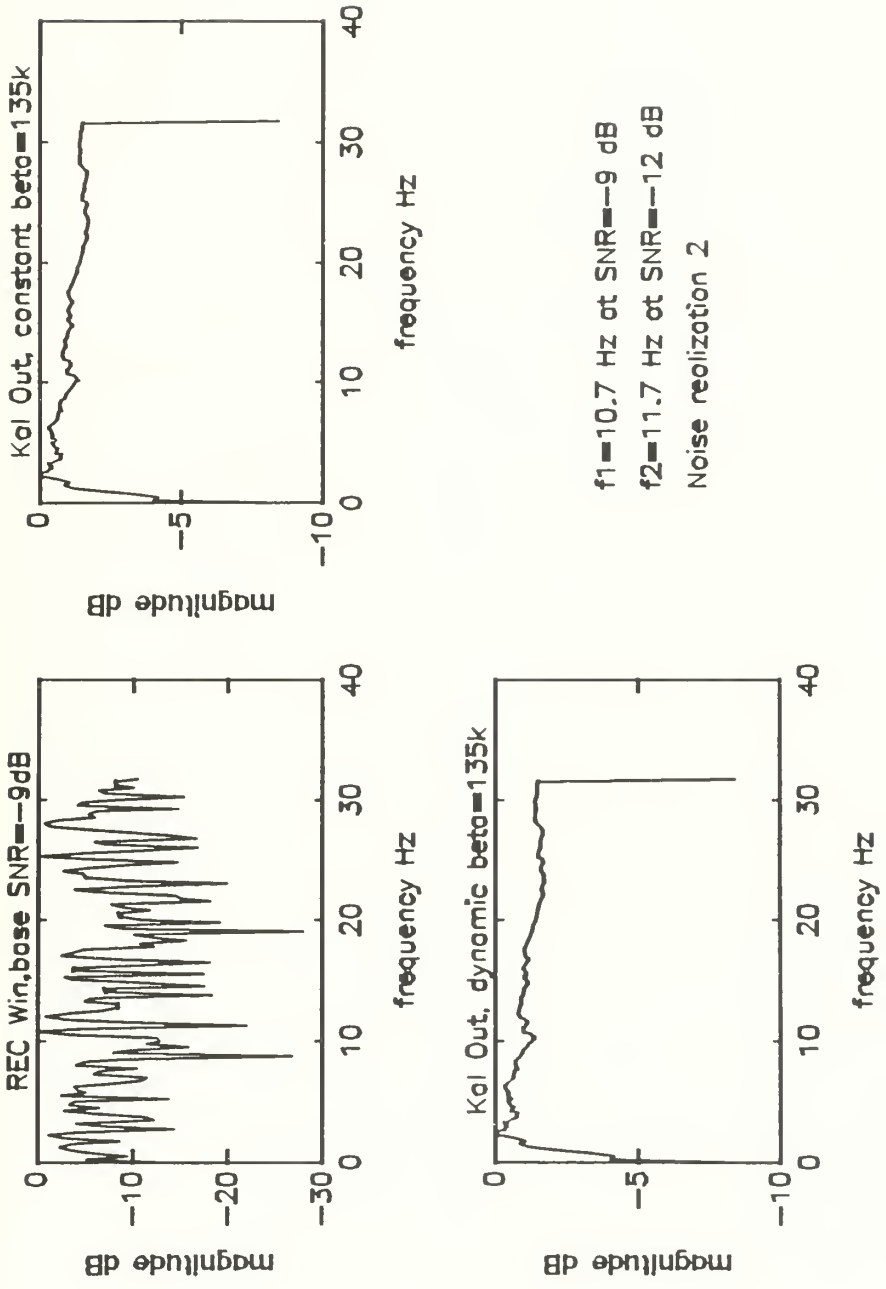

Figure 13. Periodogram, Noise Realization 2 (f1=10.7 Hz at SNR_{in}=-9 dB and f2=11.7 Hz at SNR_{in}=-12 dB). Kalman filter using a constant β and a dynamic β.

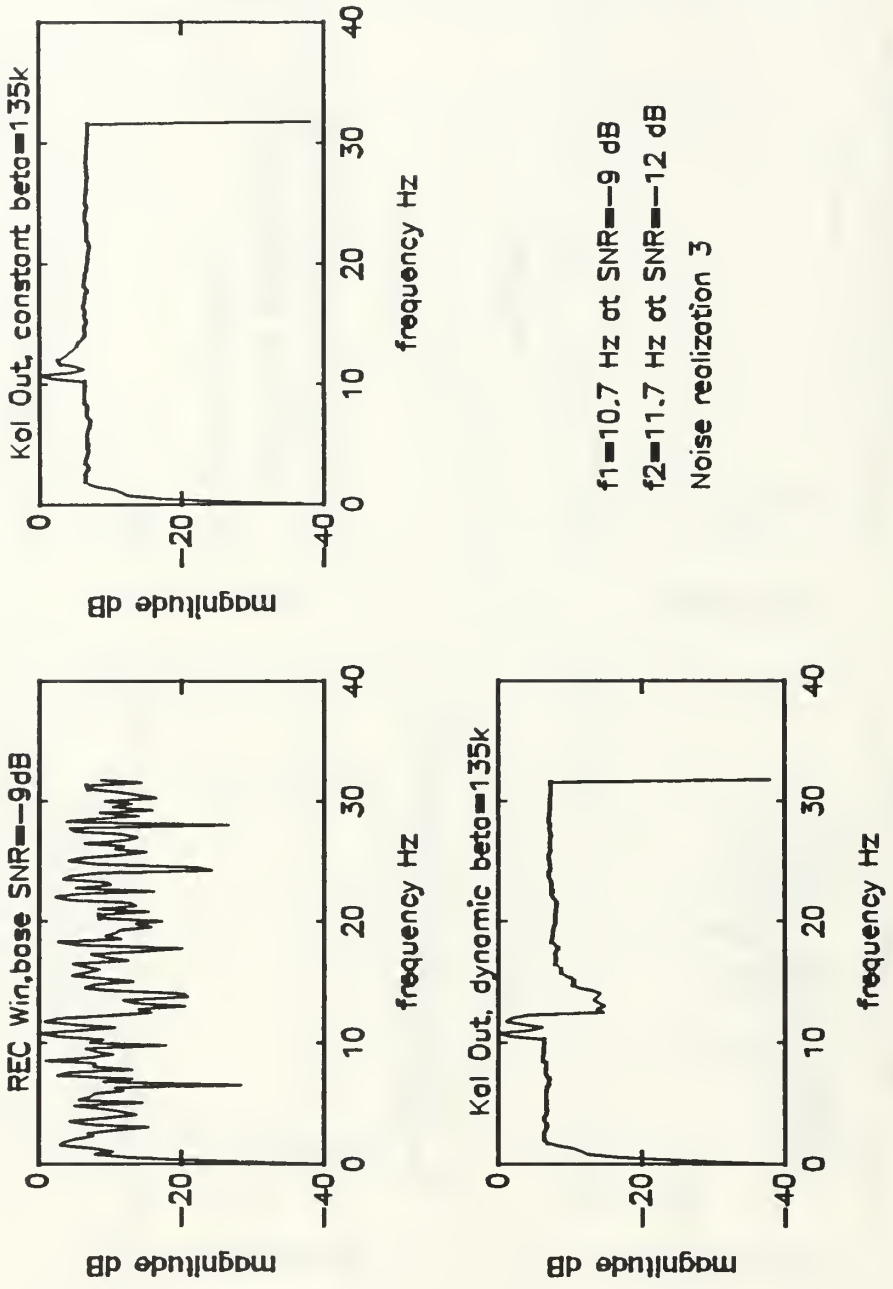

Figure 14. Periodogram, Noise Realization 3 (f1=10.7 Hz at SNR_{in}=-9 dB and f2=11.7 Hz at SNR_{in}=-12 dB). Kalman filter using a constant β and a dynamic β.

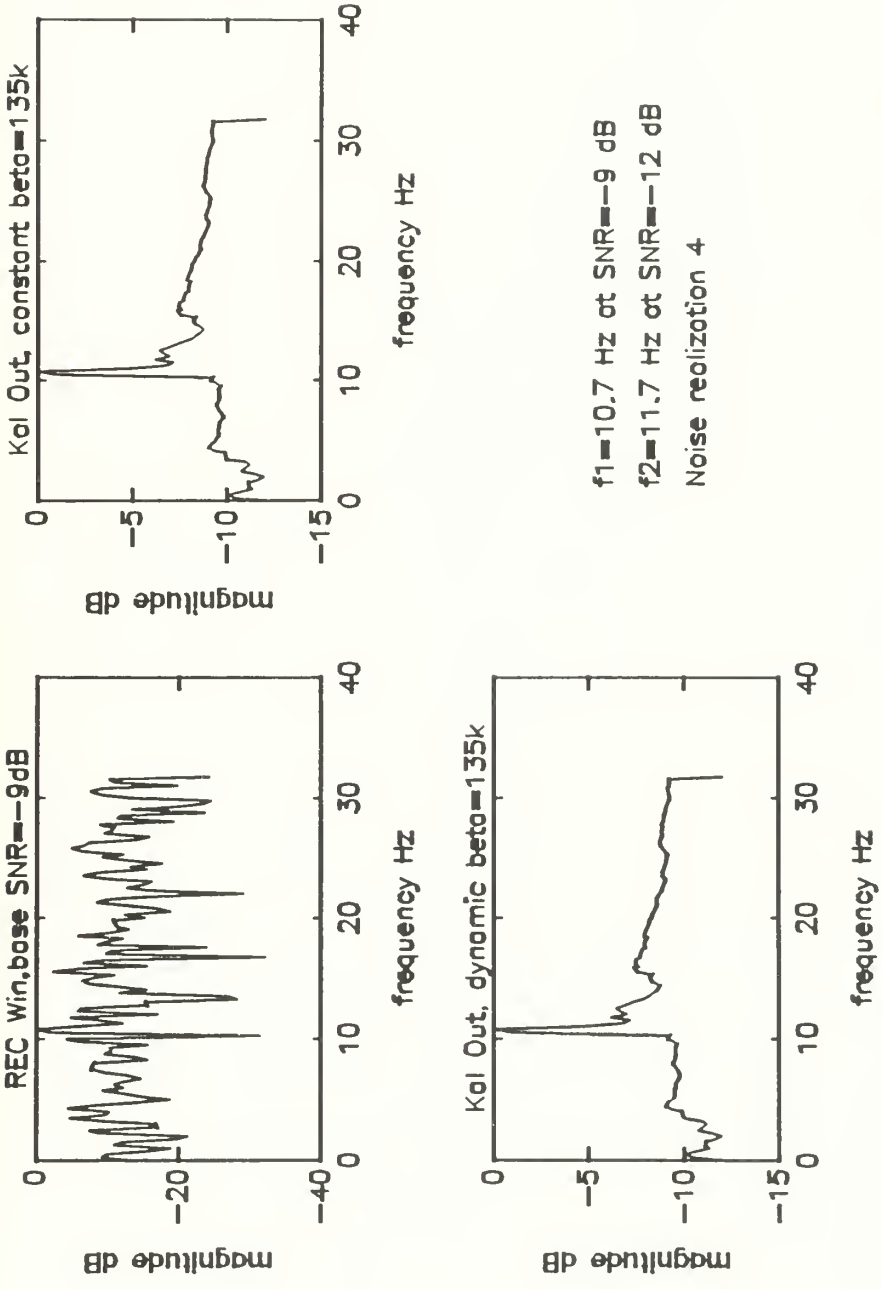

Figure 15. Periodogram, Noise Realization 4 (f1=10.7 Hz at SNR_{in}=-9 dB and f2=11.7 Hz at SNR_{in}=-12 dB). Kalman filter using a constant β and a dynamic β.

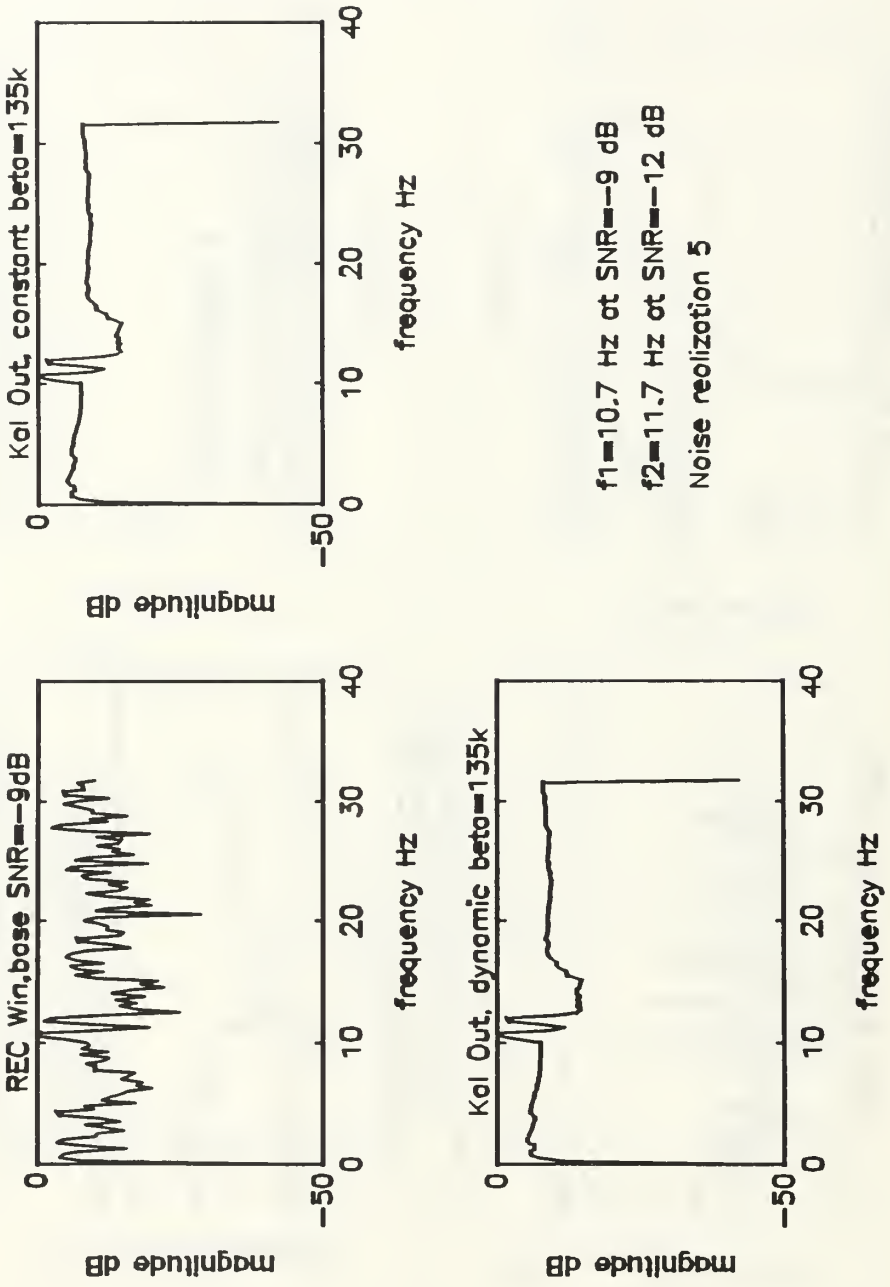

Figure 16. Periodogram, Noise Realization 5 (f1=10.7 Hz at SNR_{in}=-9 dB and f2=11.7 Hz at SNR_{in}=-12 dB). Kalman filter using a constant β and a dynamic β.

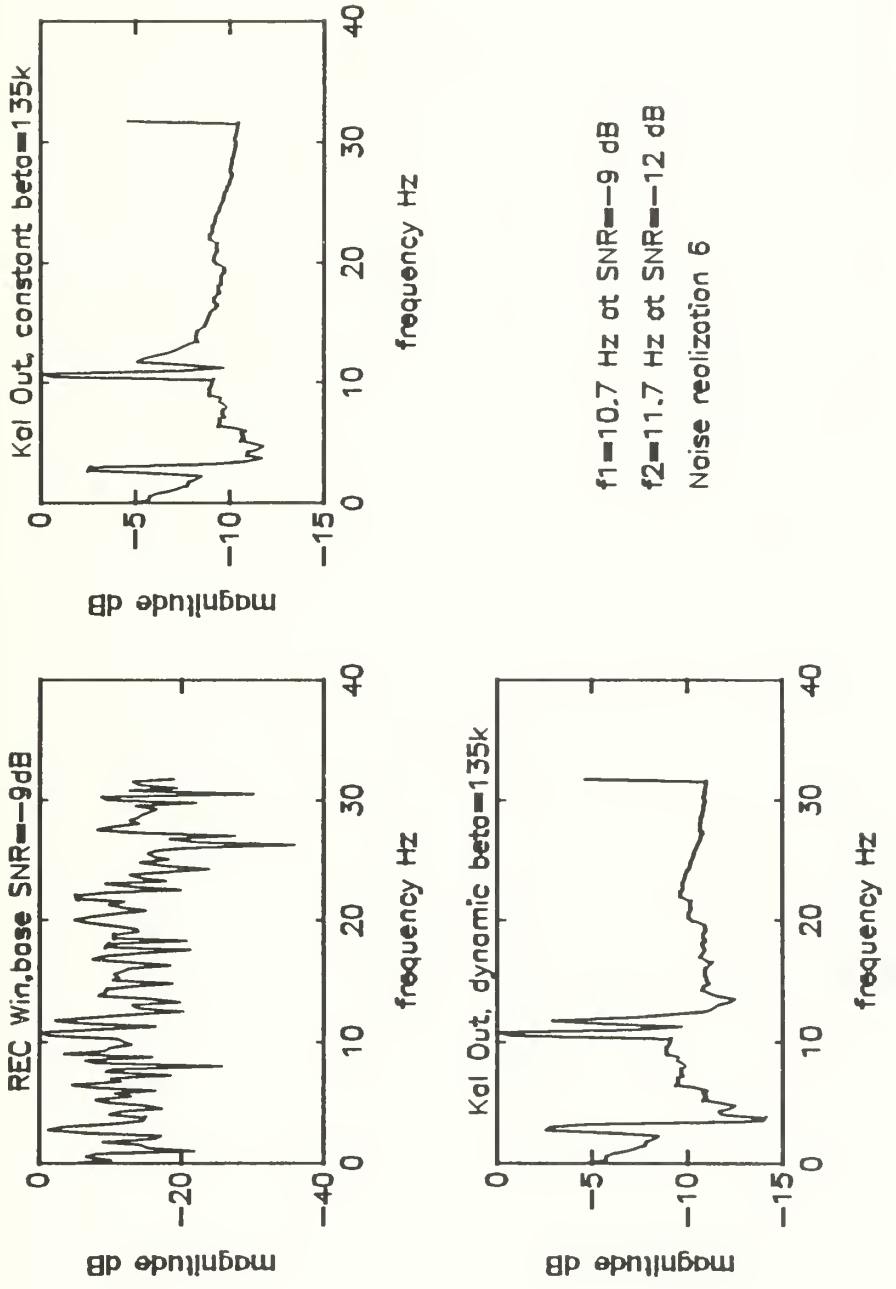

Figure 17. Periodogram, Noise Realization 6 (f1=10.7 Hz at SNR_{in}=-9 dB and f2=11.7 Hz at SNR_{in}=-12 dB). Kalman filter using a constant β and a dynamic β.

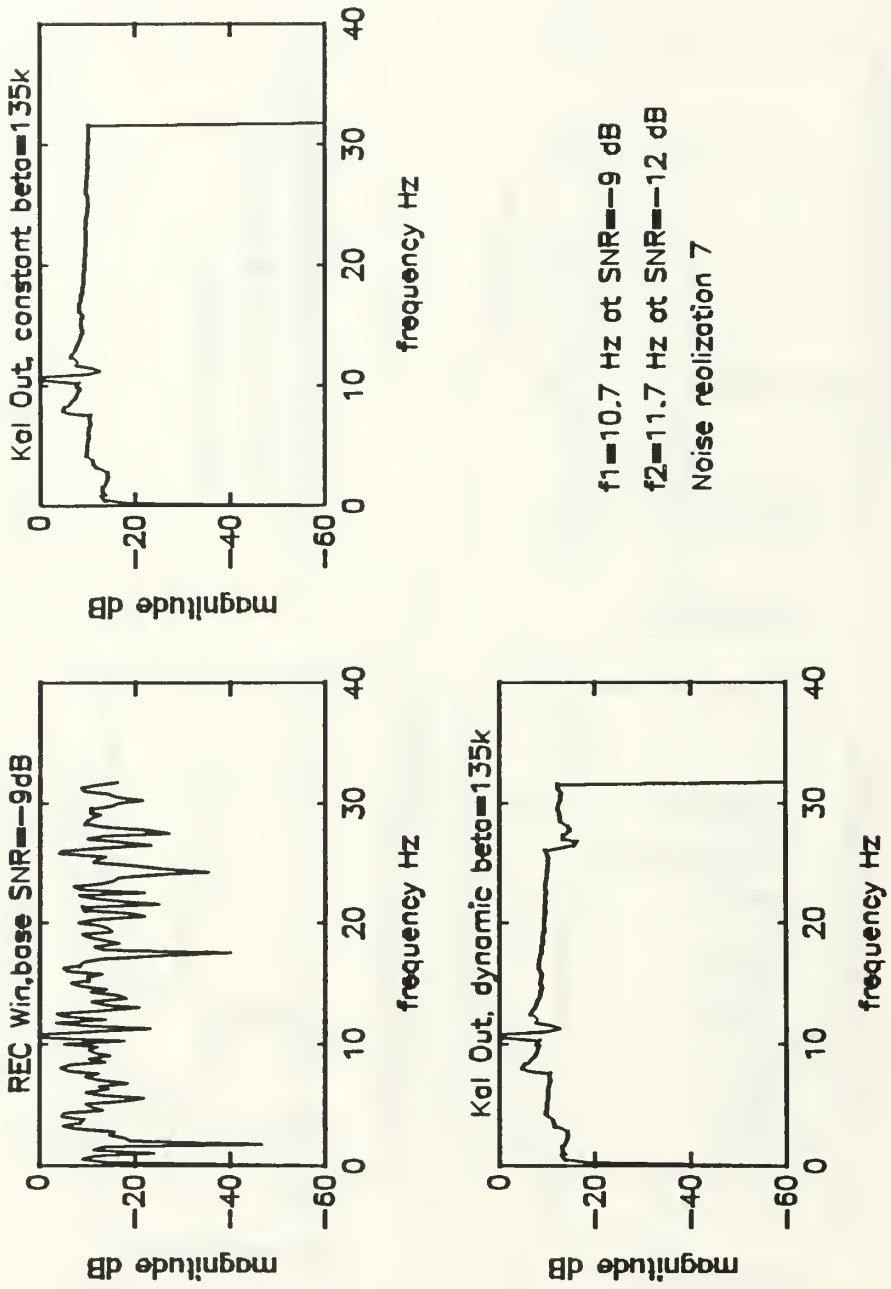

Figure 18. Periodogram, Noise Realization 7 (f1=10.7 Hz at SNR_{in}=-9 dB and f2=11.7 Hz at SNR_{in}=-12 dB). Kalman filter using a constant β and a dynamic β.

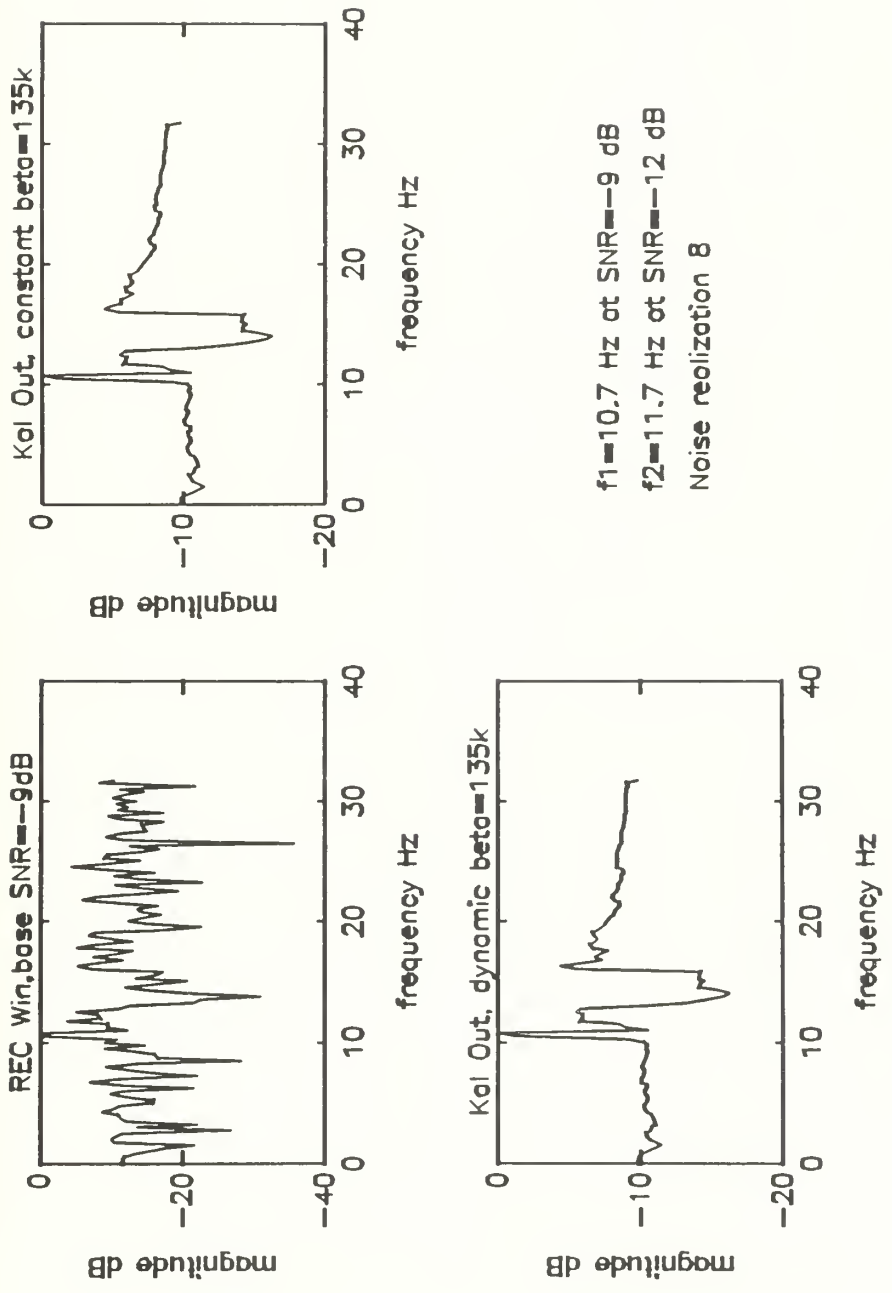

Figure 19. Periodogram, Noise Realization 8 (f1=10.7 Hz at SNR_{in}=-9 dB and f2=11.7 Hz at SNR_{in}=-12 dB). Kalman filter using a constant β and a dynamic β.

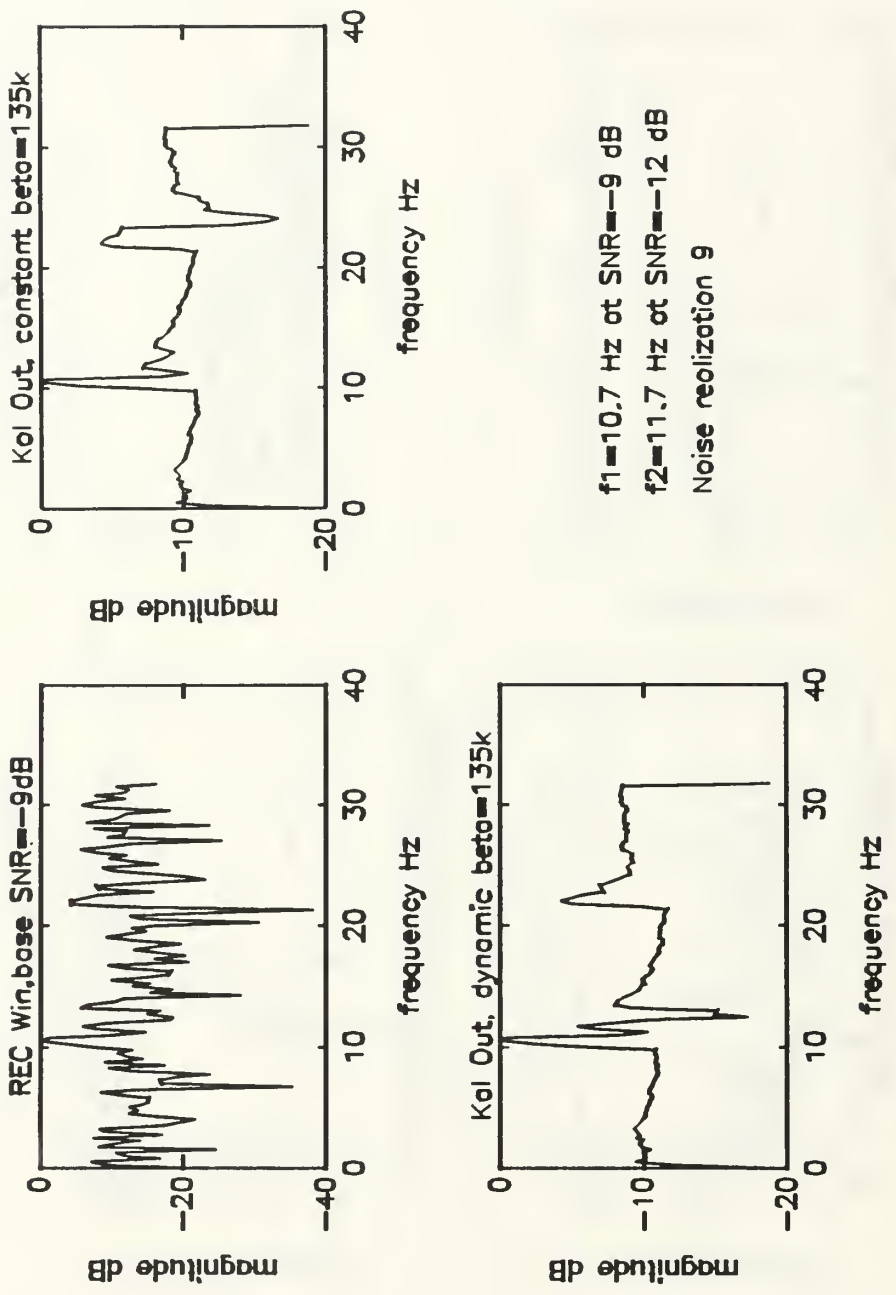

Figure 20. Periodogram, Noise Realization 9 (f1=10.7 Hz at SNR_{in}=-9 dB and f2=11.7 Hz at SNR_{in}=-12 dB). Kalman filter using a constant β and a dynamic β.

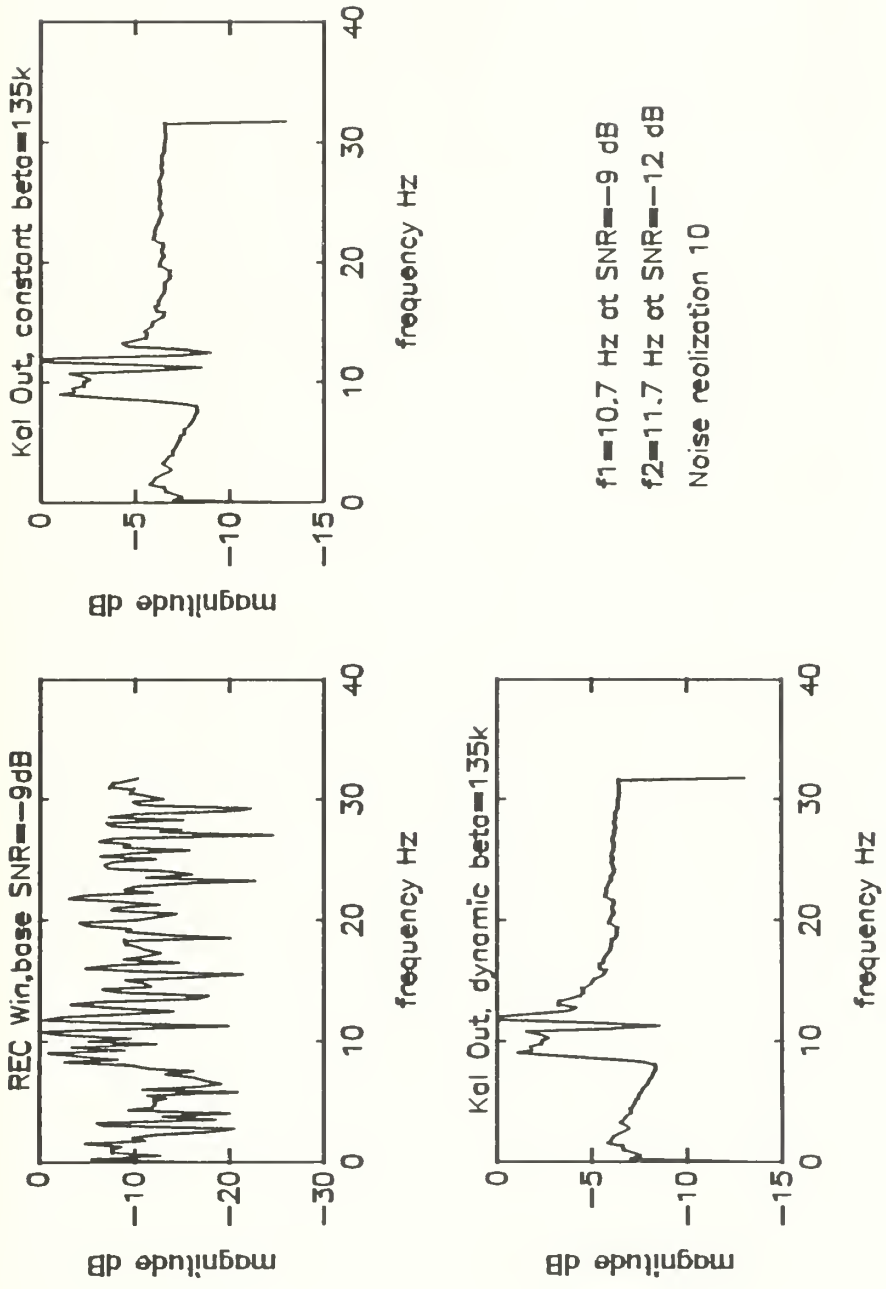

Figure 21. Periodogram, Noise Realization 10 (f1=10.7 Hz at SNR_{in}=-9 dB and f2=11.7 Hz at SNR_{in}=-12 dB). Kalman filter using a constant β and a dynamic β.

Figure 22. LOFAR presentation of a two frequency component signal (f1=10.7 Hz at SNR_{in}=-9 dB and f2=11.7 Hz at SNR_{in}=-12 dB). Kalman filter output using a constant and a dynamic β.

2. Kalman versus "Modified" Kalman

Modifying the Kalman filter to substitute raw spectral data in place of filter estimates after up transitions are detected provides two important benefits. First, in single trace outputs more detail in the vicinity of the spectral mainlobe is preserved in a large number of the noise realizations. Second, the noise floor is consistently lowered anywhere from 1 to 3 dB, with 1.5 dB appearing to be about the average improvement. This result can have a dramatic effect on the quality of the LOFAR image output. Figure 23 thru Figure 32 are 10 different noise realizations of periodograms filtered using the dynamic β and the "modified" dynamic β algorithms. The input time series contains a sinusoid at a frequency f at 10.7 Hz with a SNR_{in} of -9 dB. In 6 of the 10 noise realizations is a measurable improvement when using the "modified" dynamic β filter in that the distance of the spectral peak separation from the noise floor increases. In addition there are 3 noise realizations where the spectral resolution is better preserved by the "modified" dynamic β algorithm than by non-modified dynamic β filter algorithm.

Figure 33 shows the LOFAR output for the "modified" filter and the non-modified filter. Both filters used a dynamic β of 135000. The reduction of the noise background is obvious. The "modified" dynamic filtering scheme shows

promise to improve signal detectability as well as the frequency resolution.

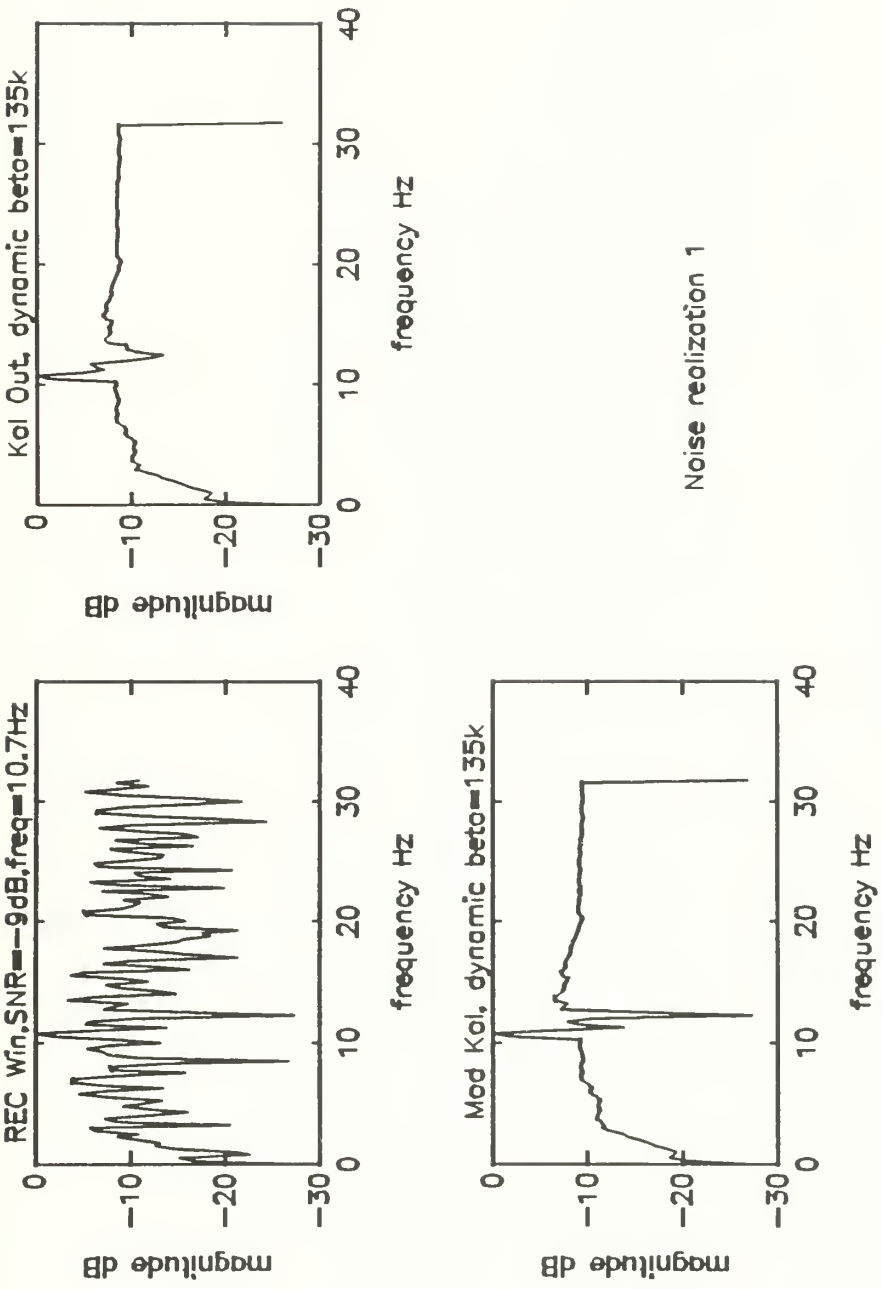

Figure 23. Periodogram, Noise Realization 1, (f1=10.7 Hz at SNR_{in}=-9 dB). Kalman filter performance using a dynamic β and "modified" dynamic β of 135000.

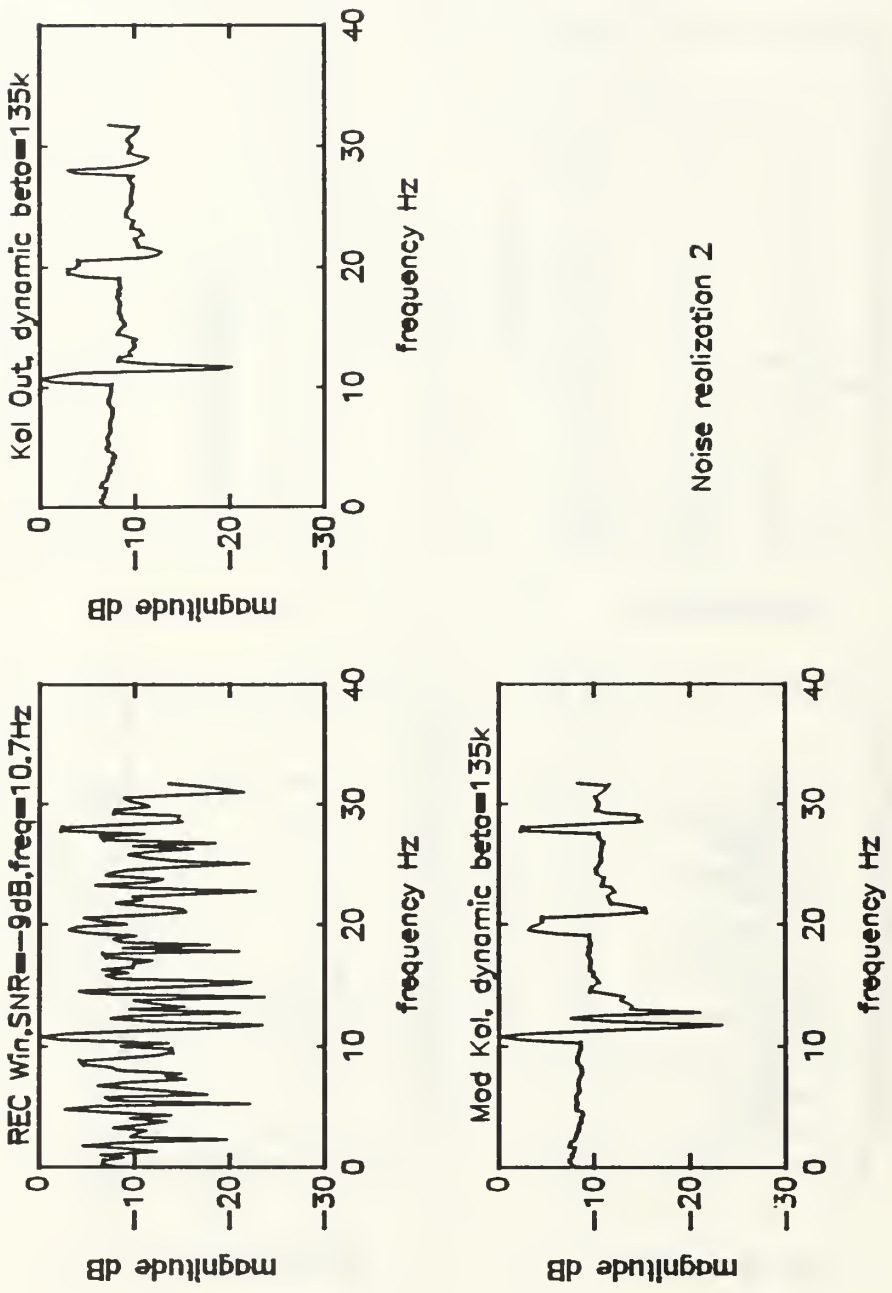

Figure 24. Periodogram, Noise Realization 2, (f1=10.7 Hz at SNR_{in}=-9 dB). Kalman filter performance using a dynamic β and "modified" dynamic β of 135000.

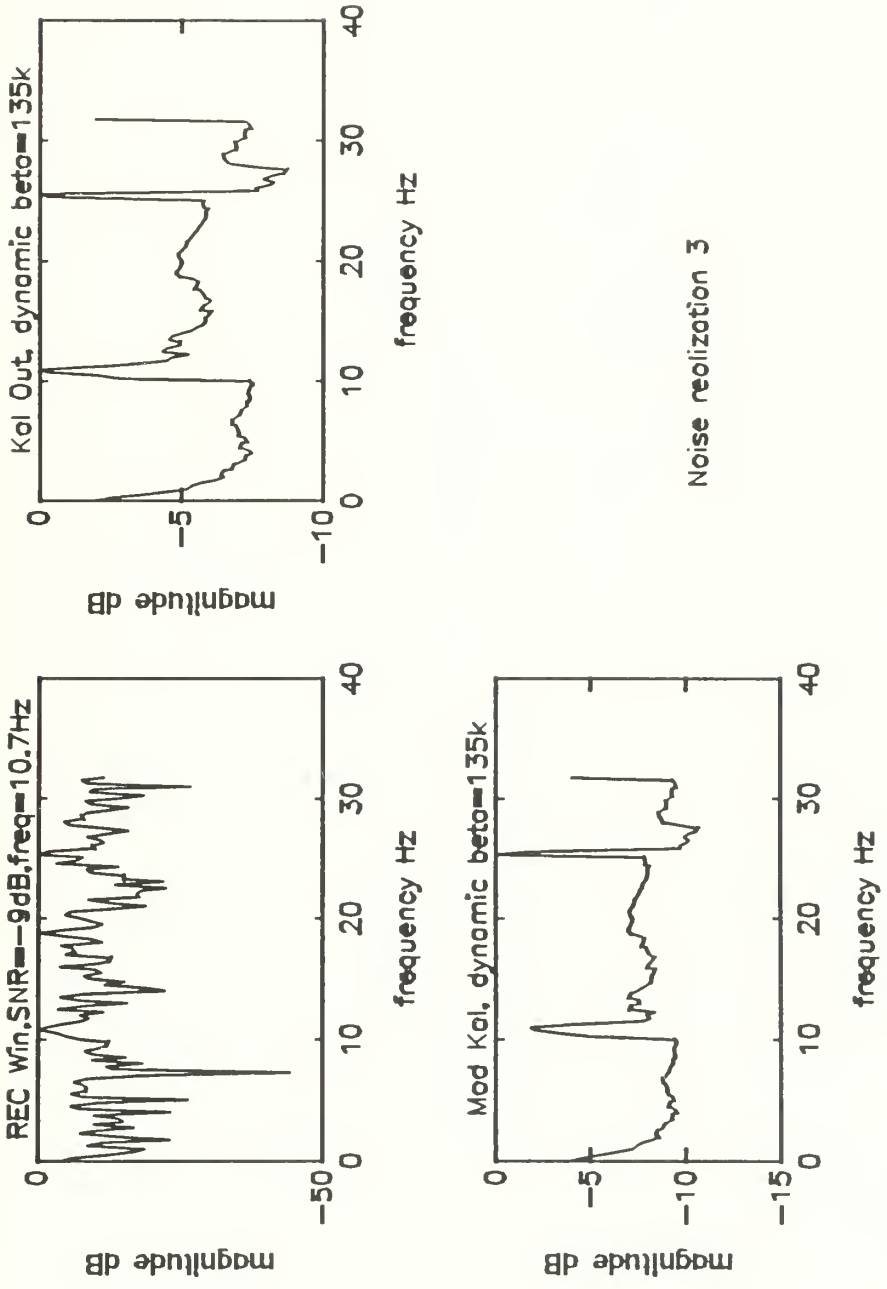

Figure 25. Periodogram, Noise Realization 3, (f1=10.7 Hz at SNR_{in}=-9 dB). Kalman filter performance using a dynamic β and "modified" dynamic β of 135000.

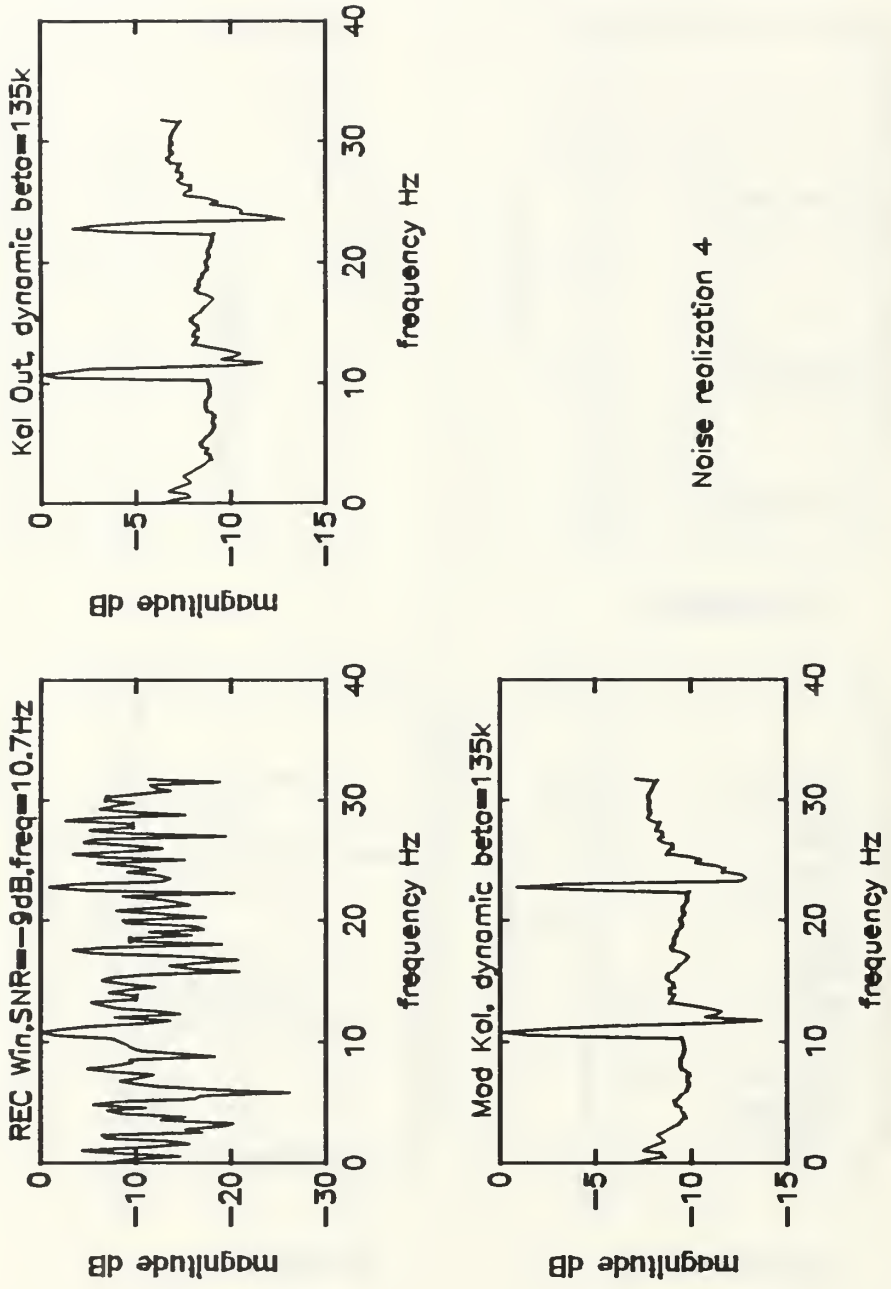

Figure 26. Periodogram, Noise Realization 4, (f1=10.7 Hz at SNR_{in}=-9 dB). Kalman filter performance using a dynamic β and "modified" dynamic β of 135000.

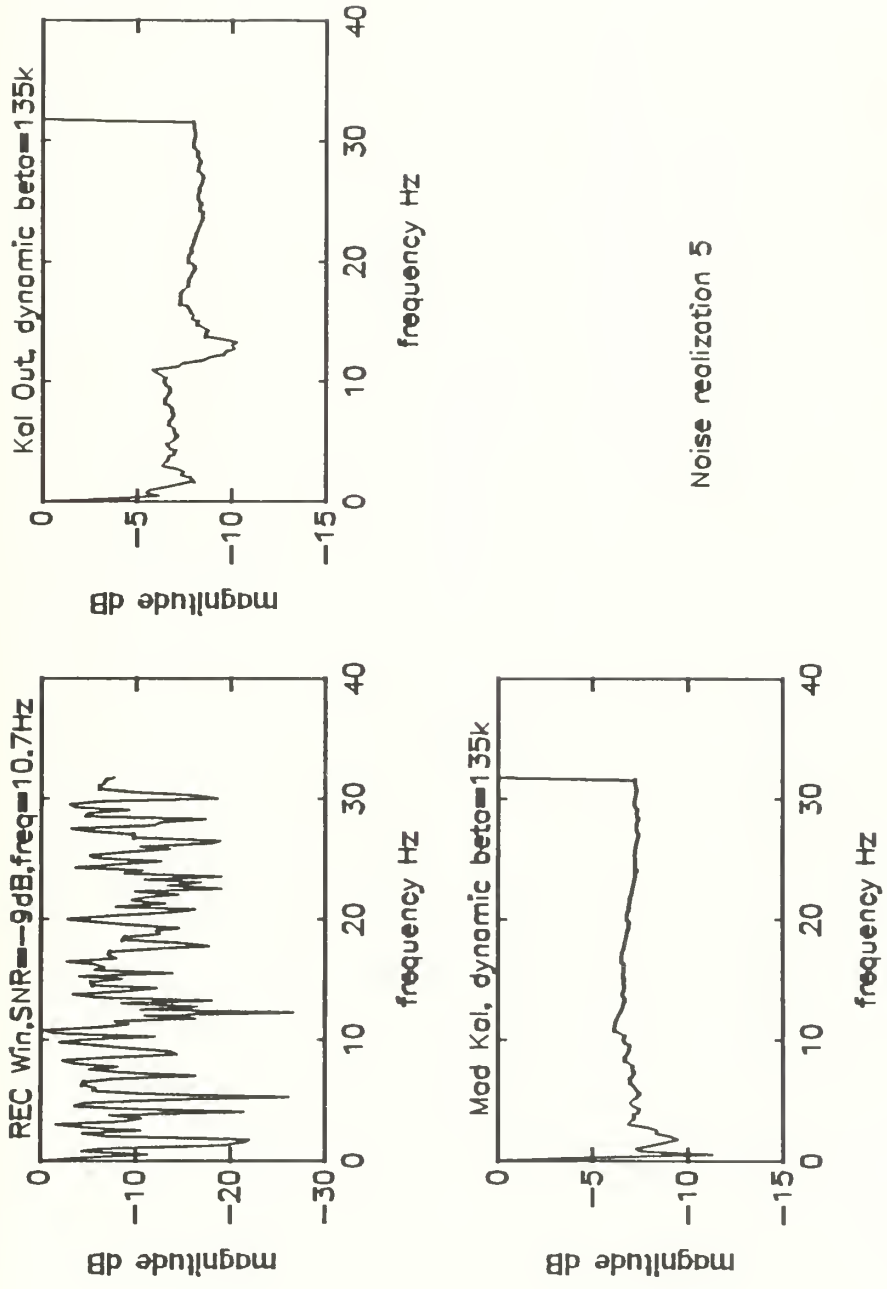

Figure 27. Periodogram, Noise Realization 5, (f1=10.7 Hz at SNR_{in}=-9 dB). Kalman filter performance using a dynamic β and "modified" dynamic β of 135000.

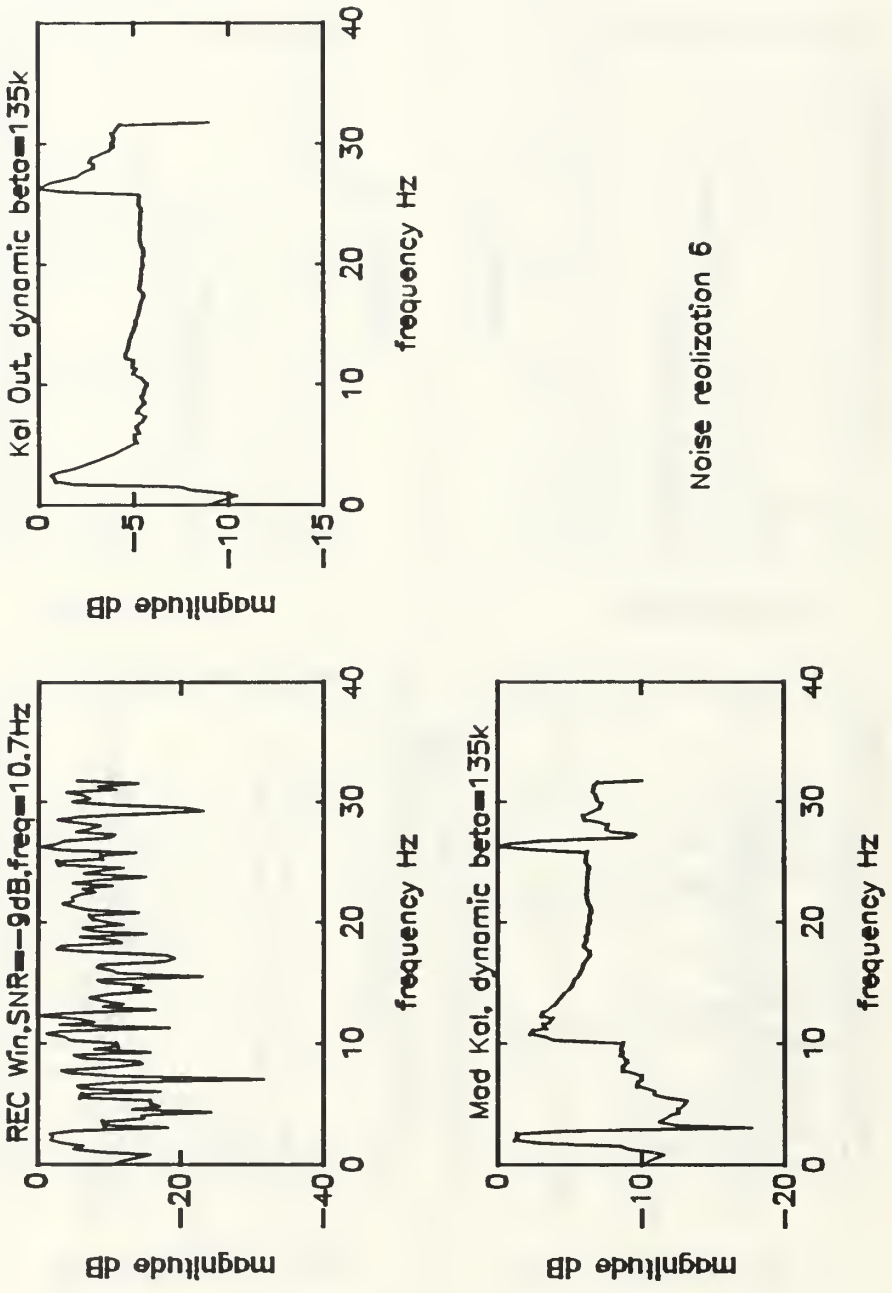

Figure 28. Periodogram, Noise Realization 6, (f1=10.7 Hz at SNR_{in}=-9 dB). Kalman filter performance using a dynamic β and "modified" dynamic β of 135000.

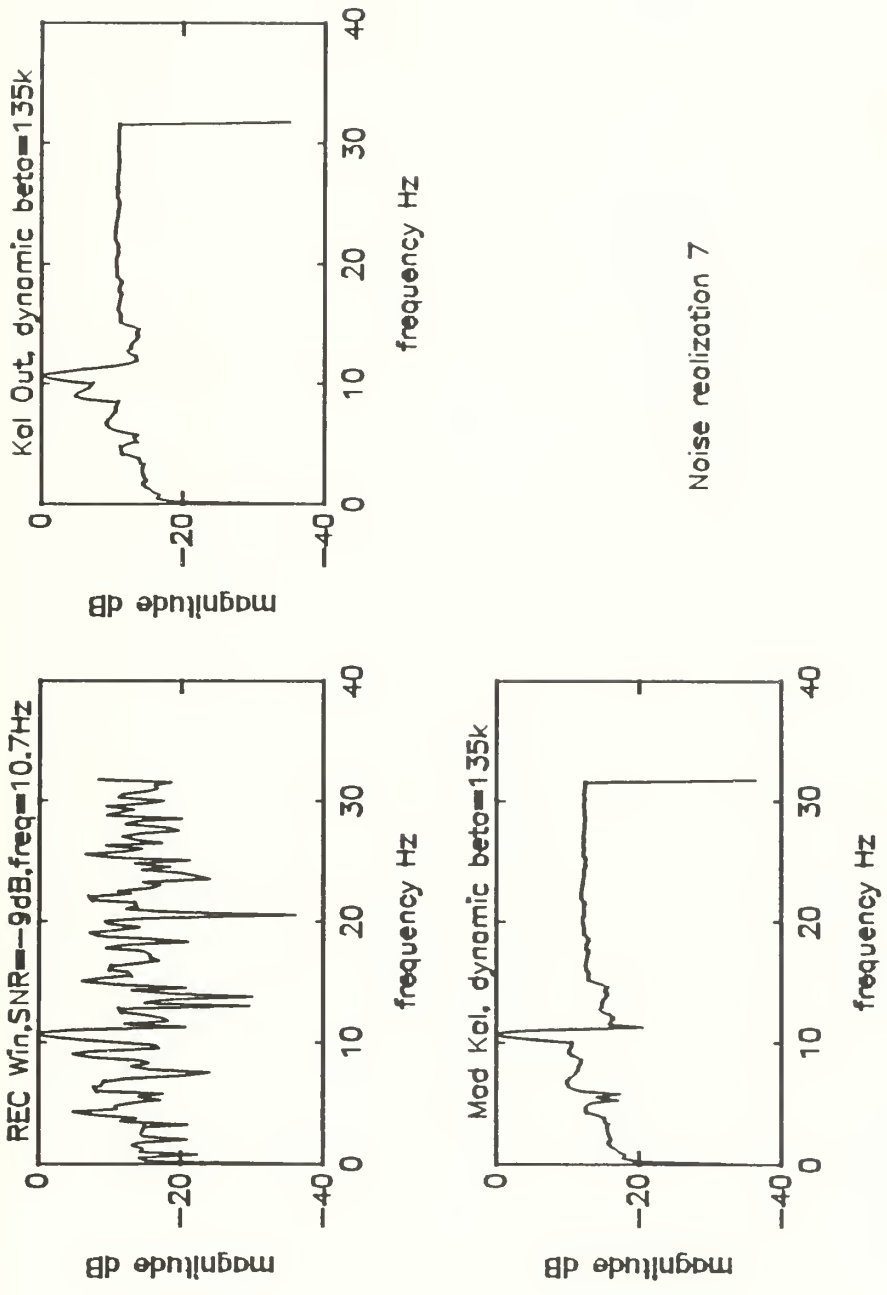

Figure 29. Periodogram, Noise Realization 7, (f1=10.7 Hz at SNR_{in}=-9 dB). Kalman filter performance using a dynamic β and "modified" dynamic β of 135000.

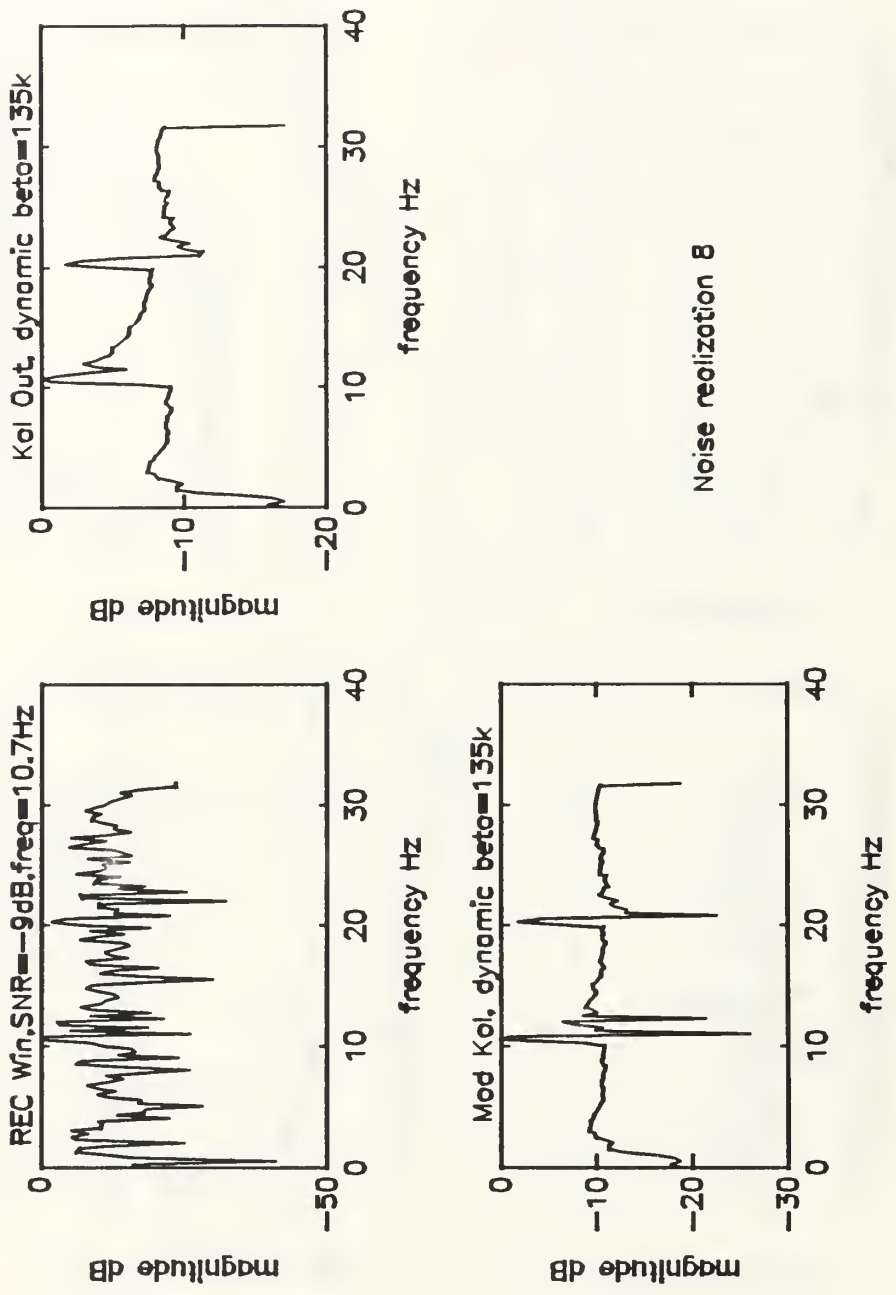

Figure 30. Periodogram, Noise Realization 8, (f1=10.7 Hz at SNR_{in}=-9 dB). Kalman filter performance using a dynamic β and "modified" dynamic β of 135000.

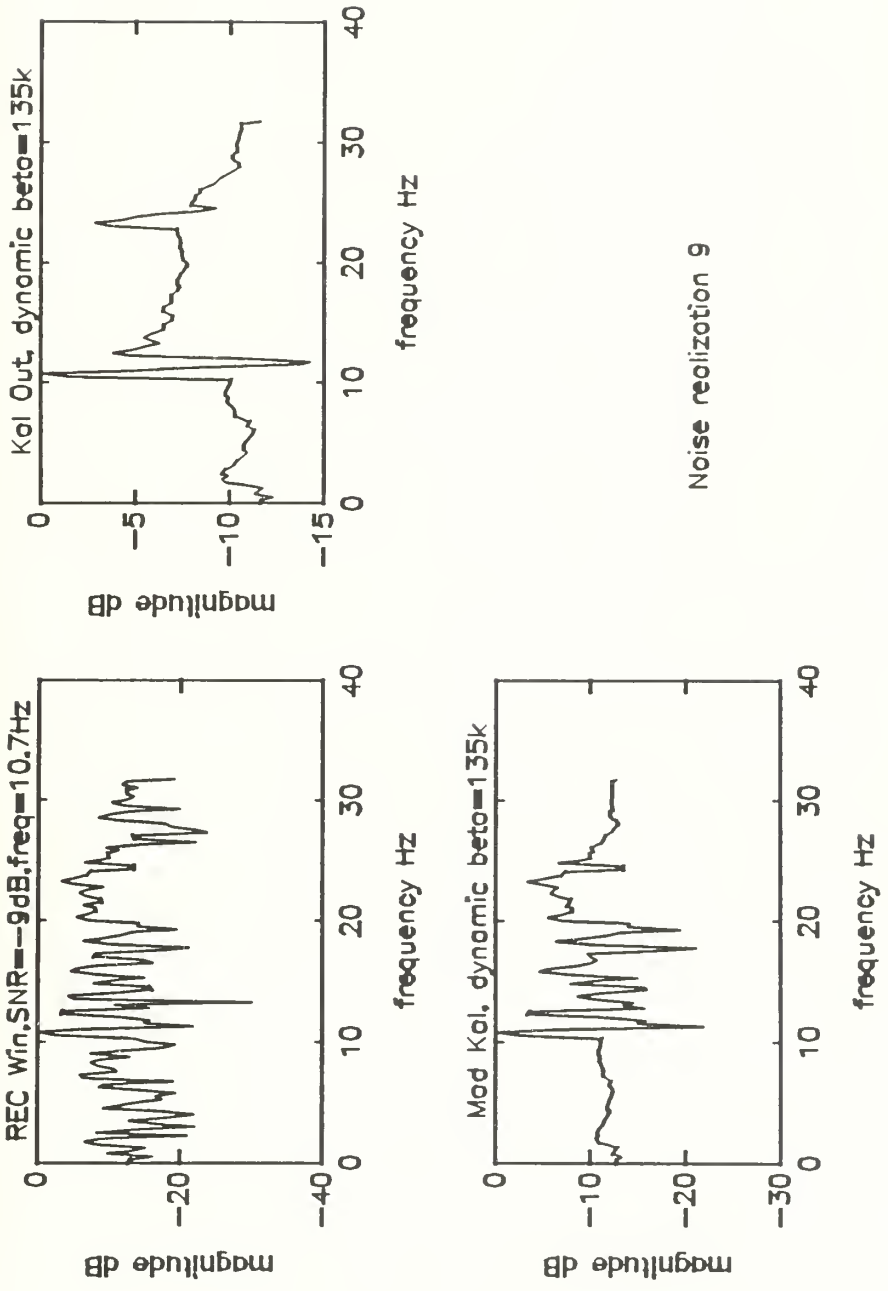

Figure 31. Periodogram, Noise Realization 9, (f1=10.7 Hz at SNR_{in}=-9 dB). Kalman filter performance using a dynamic β and "modified" dynamic β of 135000.

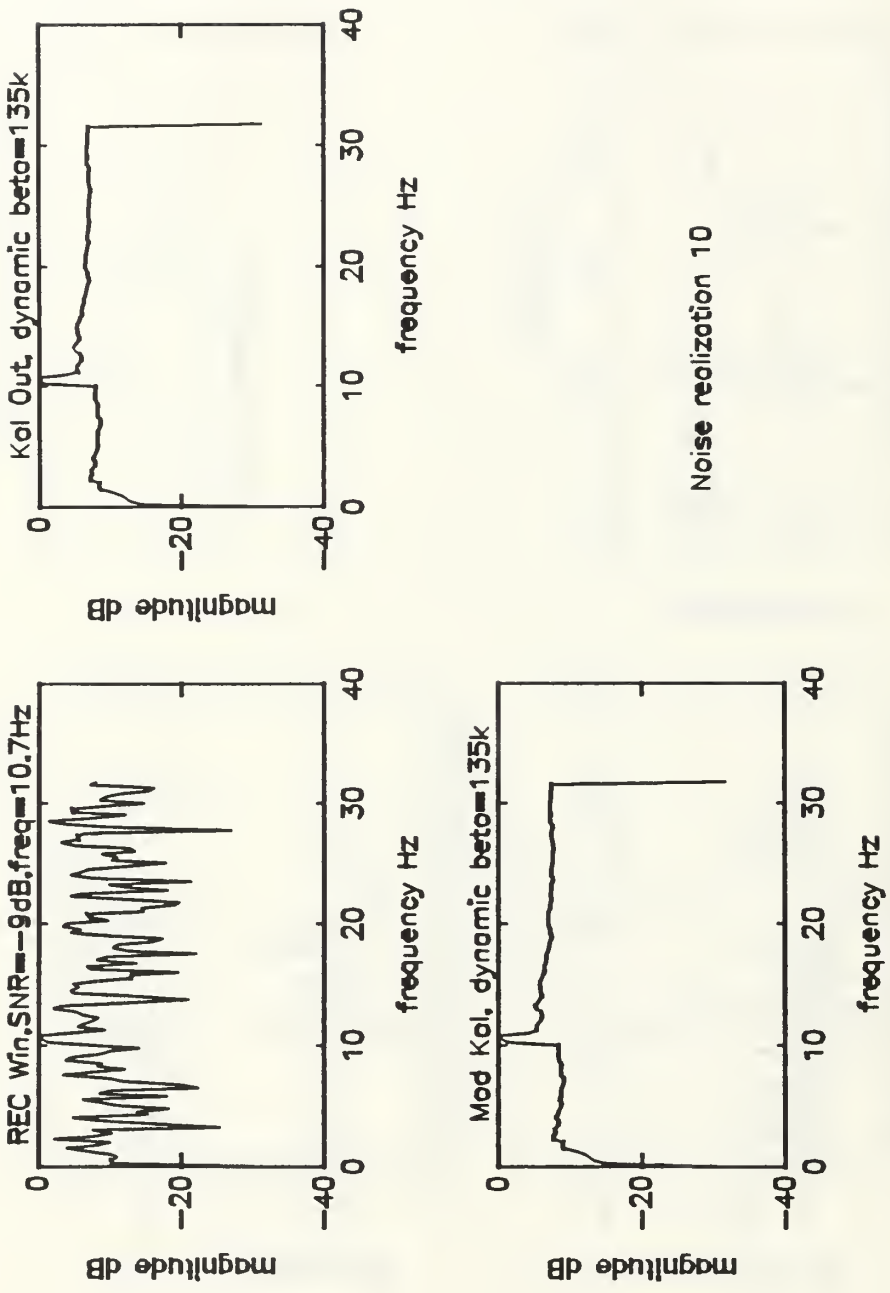

Figure 32. Periodogram, Noise Realization 10, (f1=10.7 Hz at SNR_{in}=-9 dB). Kalman filter performance using a dynamic β and "modified" dynamic β of 135000.

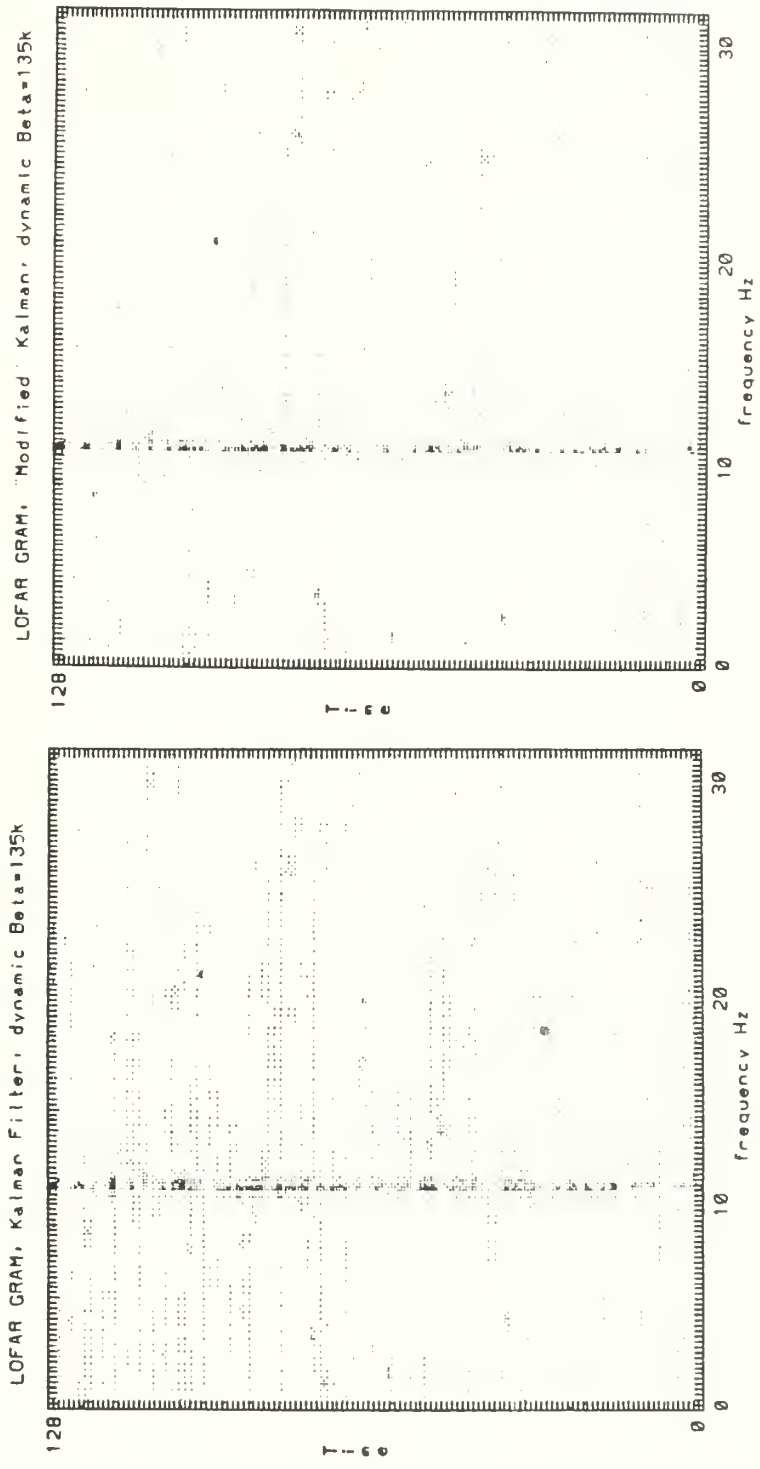

Figure 33. LOFAR output (f1=10.7 Hz at SNR_{in}=-9 dB). Kalman filter output using a dynamic β and a "modified" dynamic β of 135000.

B. "MODIFIED" KALMAN FILTERING VERSUS HAMMING WINDOWING

It has been demonstrated that the Kalman filter more accurately preserves the spectral resolution of the original periodogram whereas the Hamming window will broaden the mainlobe (see Figure 6). The statistics tabulated in Table II (p. 40) clearly show that the Kalman filter (β=80k and 135k for both the "modified" dynamic β and the dynamic β) is more successful at signal detection at an SNR_{in} of -9 and of -12 dB than the Hamming window.

Figure 34 displays the LOFAR output of a 3 component sinusoidal signal (f1=10.7 Hz at SNR_{in}=-9 dB, f2=14.45 Hz at SNR_{in}=-15 dB and f3=20.25 Hz at SNR_{in}=-9 dB) of rectangularly windowed segments. Figure 35 permits the comparison of using the Hamming window and the "modified" Kalman filter with a dynamic β of 135k on the same data.

The "modified" Kalman filter preserved the spectral resolution of the original unwindowed spectra where as the mainlobe broadening is evident when applying the Hamming window. This broadening can have a major impact in detecting nearby spectral components. See Appendix B for a closer look at resolution comparisons between Hamming window and the "modified" Kalman filter. Also note the reduction in the noise background of the Kalman LOFAR output which enhances the -15 dB component more clearly than the Hamming window.

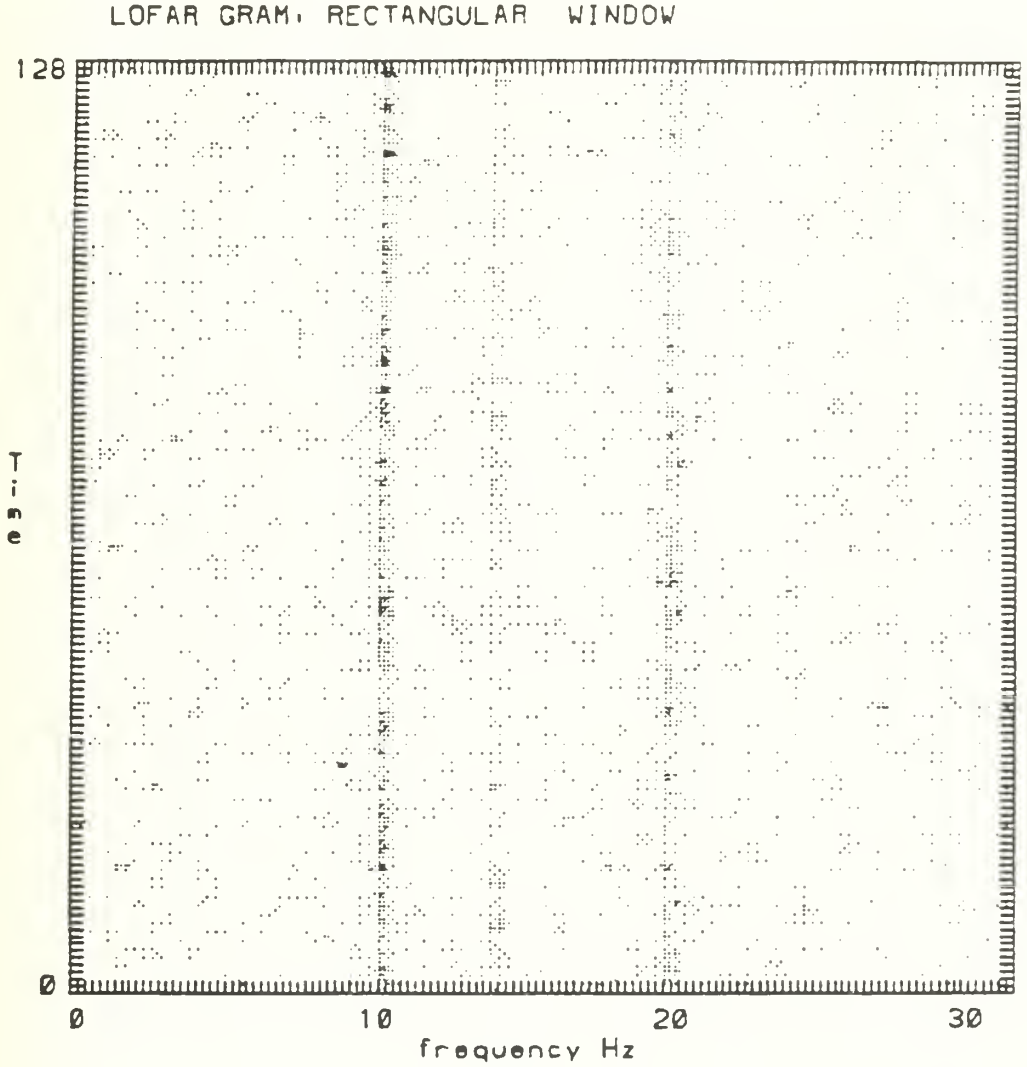

Figure 34. LOFAR output of Rectangular windowed data of 3 component signal (f1=10.7 Hz at SNR_{in}=-9 dB, f2=14.45 Hz at SNR_{in}=-15 dB and f3=20.25 Hz at SNR_{in}=-12 dB).

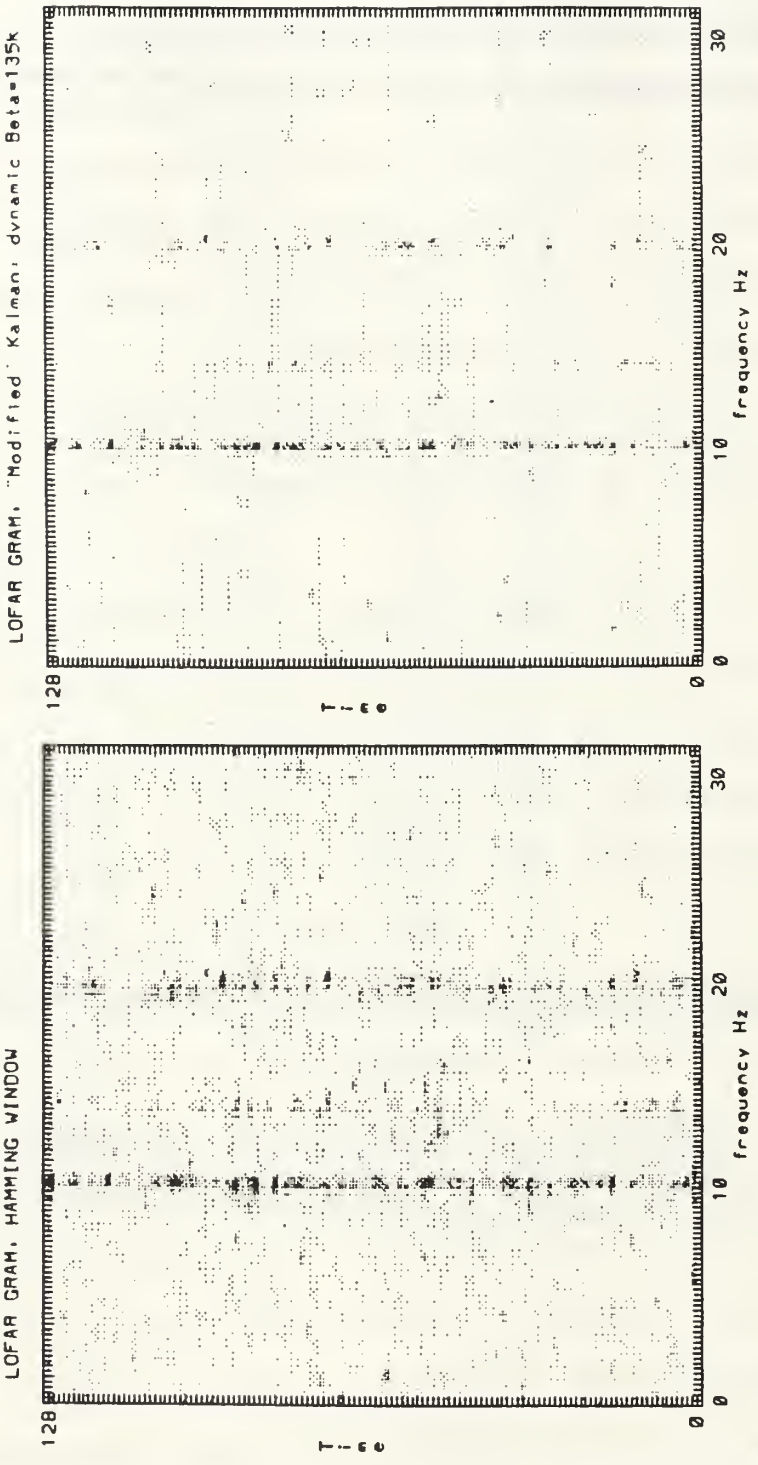

Figure 35. LOFAR output of the 3 component signal of Figure 34. Hamming window and Kalman ("modified" dynamic β=135k) filter.

C. EFFECTS OF SNR_{out}

Kalman filtering consistently permits detection of signals corrupted by additive Gausian white noise at a SNR_{out} of 6 dB or greater. Consider the two extreme cases of signal only and noise only. A signal at SNR_{in}=30 dB (signal only case) filtered using the "modified" Kalman filter (using a dynamic β=8k, 80k and 800k) produces excellent results as shown in Figure 36 where the "modified" Kalman filter detects the frequency component and preserves the resolution regardless of the β used. In the case of noise only (SNR_{in}=-60 dB) the "modified" Kalman filter output provided in Figure 37 is identical for all three β's used in processing the original periodograms.

To take a closer look at the "modified" Kalman filter performance with SNR_{out} of less than 9 dB, we consider a signal of 6 non-equally spaced frequency components at f1=3.3 Hz with SNR_{in}=-9 dB, f2=8.7 Hz with SNR_{in}=-12 dB, f3=14.4 Hz with SNR_{in}=-13 dB, f4=19.3 Hz with SNR_{in}=-14 dB, f5=23.7 Hz with SNR_{in}=-15 dB and f6=28.3 Hz with SNR_{in}=-16 dB. We filter this signal with a Hamming window and a "modified" Kalman filter using a dynamic β=80k. Figure 38 displays the LOFAR outputs of these filters. In both LOFAR outputs f1, f2 and f3 are detectable. However, note the reduced noise background in the "modified" Kalman filter and the superior resolution of the "modified" Kalman filter (most obvious for f1). The tonal line of f4

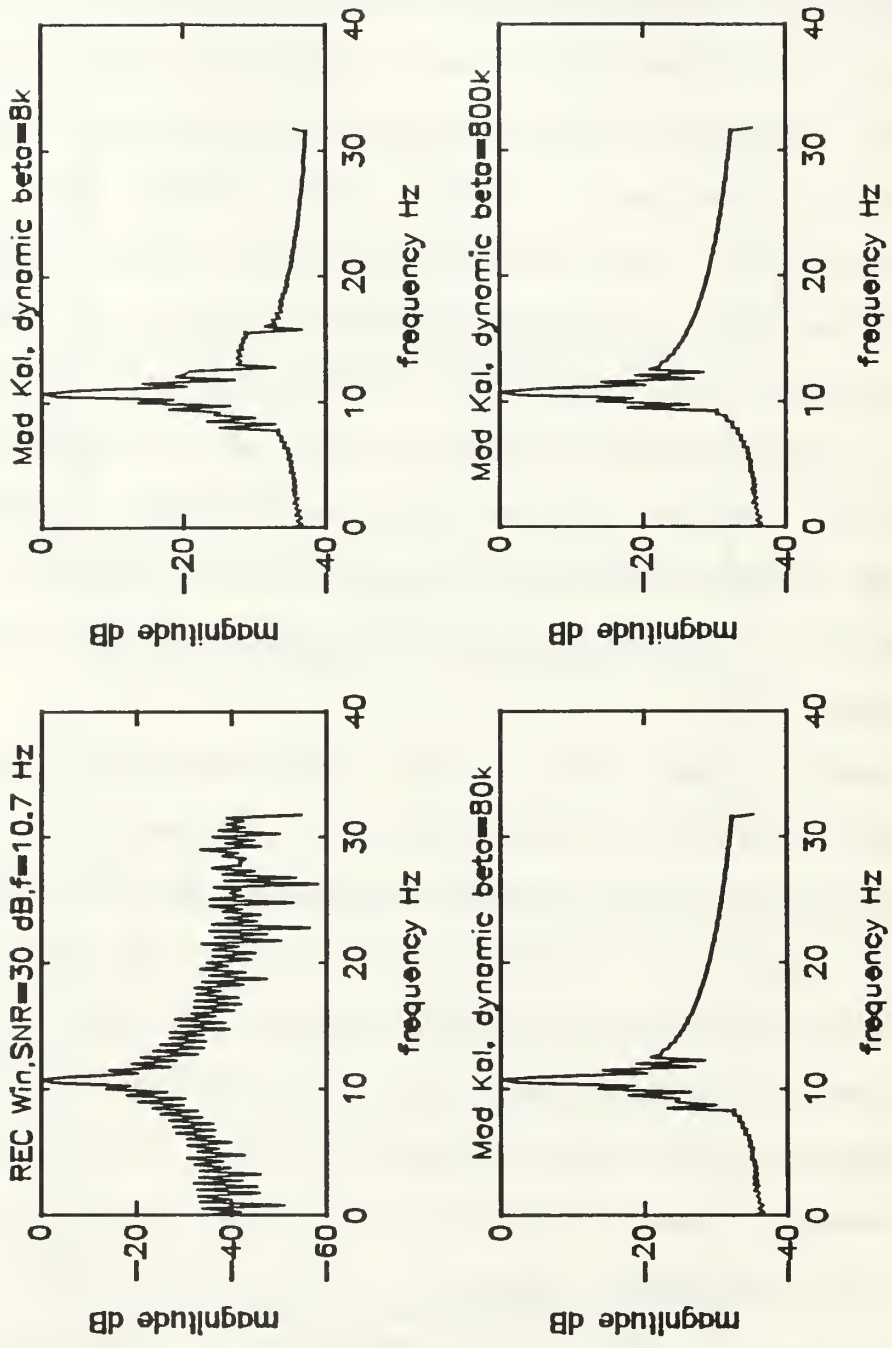

Figure 36. Periodogram of f=10.7 Hz at SNR_{in}=30 dB (signal only) filtered using a Kalman ("modified" dynamic β) filter for β=8k, β=80k and β=800k.

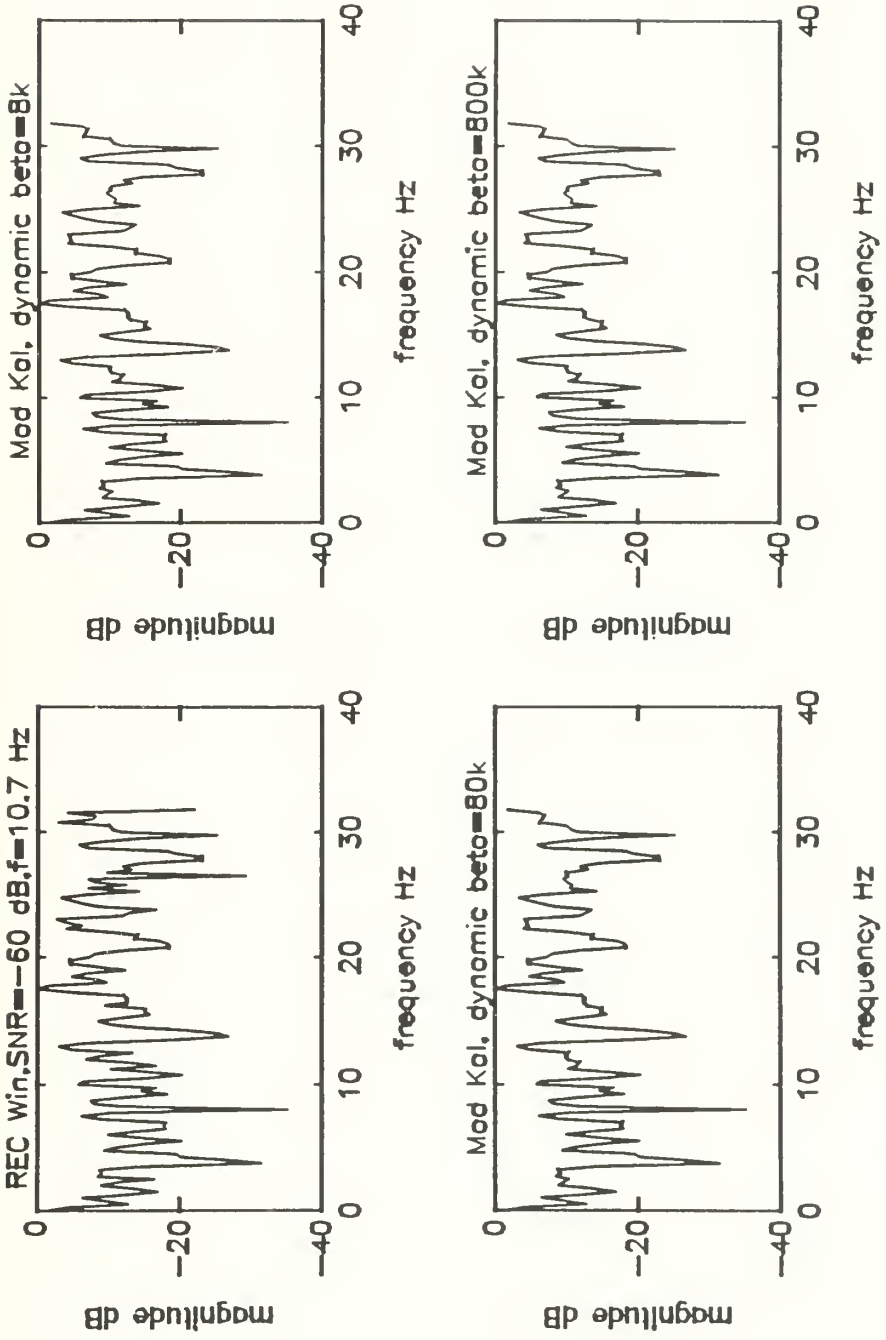

Figure 37. Periodogram of f=10.7 Hz at SNR_{in}=-60 dB (noise only) filtered using a Kalman ("modified" dynamic β) filter for β=8k, β=80k and β=800k.

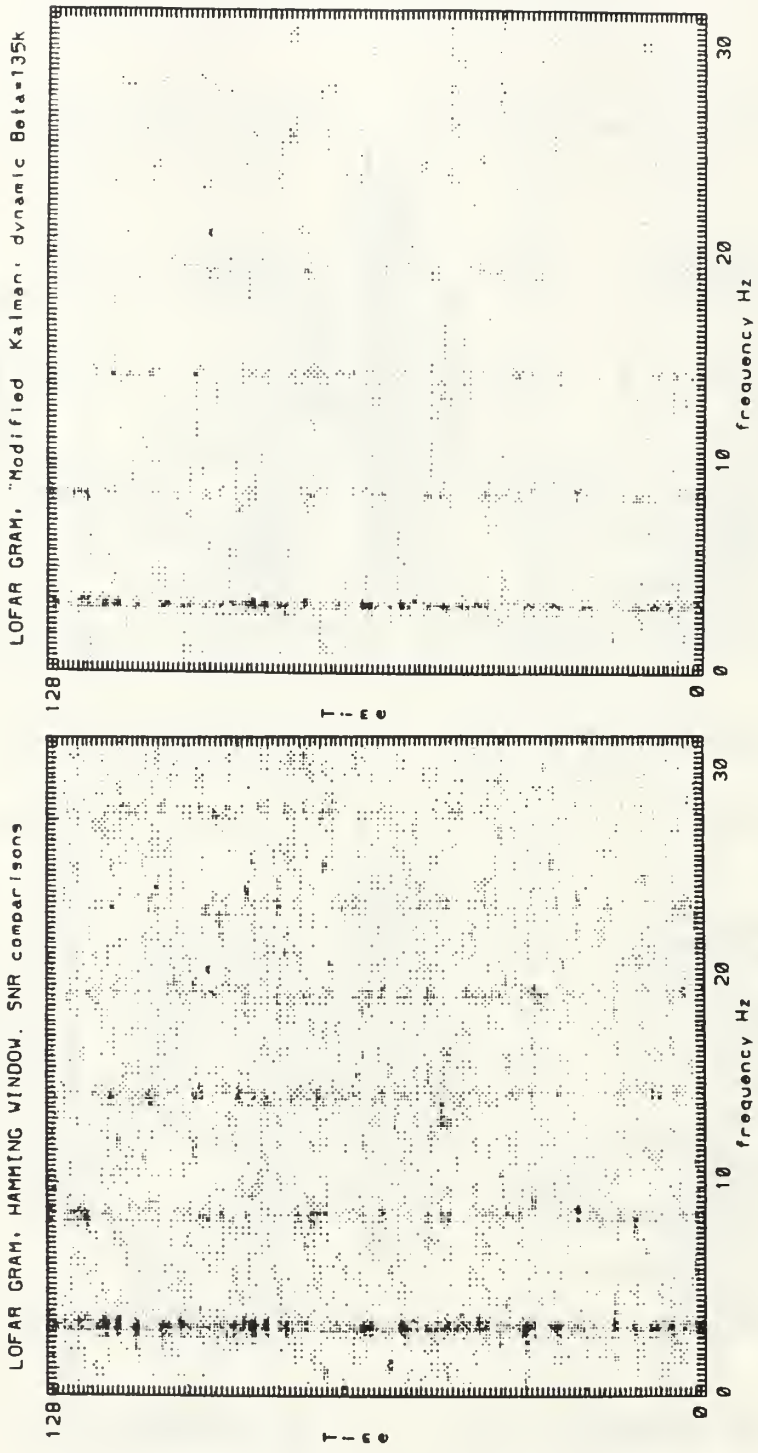

Figure 38. LOFAR outputs using Hamming window and "modified" Kalman (dynamic β=135k). Freqs(Hz) at $SNRs_{in}$(dB); f1=3.3 at -9, f2=8.7 at -12, f3=14.4 at -13, f4=19.3 at -14, f5=23.7 at -15 and f6=28.3 at -16.

(SNR_{in}=-14 dB) is also present in both but is not nearly as distinct. It is much more difficult to observe f5 or f6 using either filtering process.

Further processing gain (for stationary frequency components) can be realized by averaging the columns of the periodogram arrays. The procedure used in this study is to sum the columns and divide the sum by the number of rows (128 in our case). Let $Y(j,k)$ be an input array of size J by N then $X(k)$ is calculated as follows;

$$SumY(k) = \sum_{j=1}^{J} \frac{Y(j,k)}{J}$$

$$SumYnorm(k) = \frac{SumY(k)}{\max(SumY(k))}$$

$$X(k) = 10 \cdot \log_{10}(SumYnorm(k)) \quad k=1,2,\ldots,N. \qquad (25)$$

Figure 39 is the output of this process for the rectangular window, Hamming window, Kalman ("modified" dynamic β=80k) and Kalman ("modified" dynamic β=135k). All six frequency components are clearly present in all the displays. It appears however that Kalman is superior in smoothing the noise variance between components.

To take this experiment one step further a new signal was generated using the same frequencies as before but at lower SNR's (f1 at SNR_{in}=-16 dB, f2 at SNR_{in}=-17 dB, f3 at SNR_{in}=-18 dB, f4 at SNR_{in}=-19 dB, f5 at SNR_{in}=-20 dB and f6 at SNR_{in}=-21 dB).

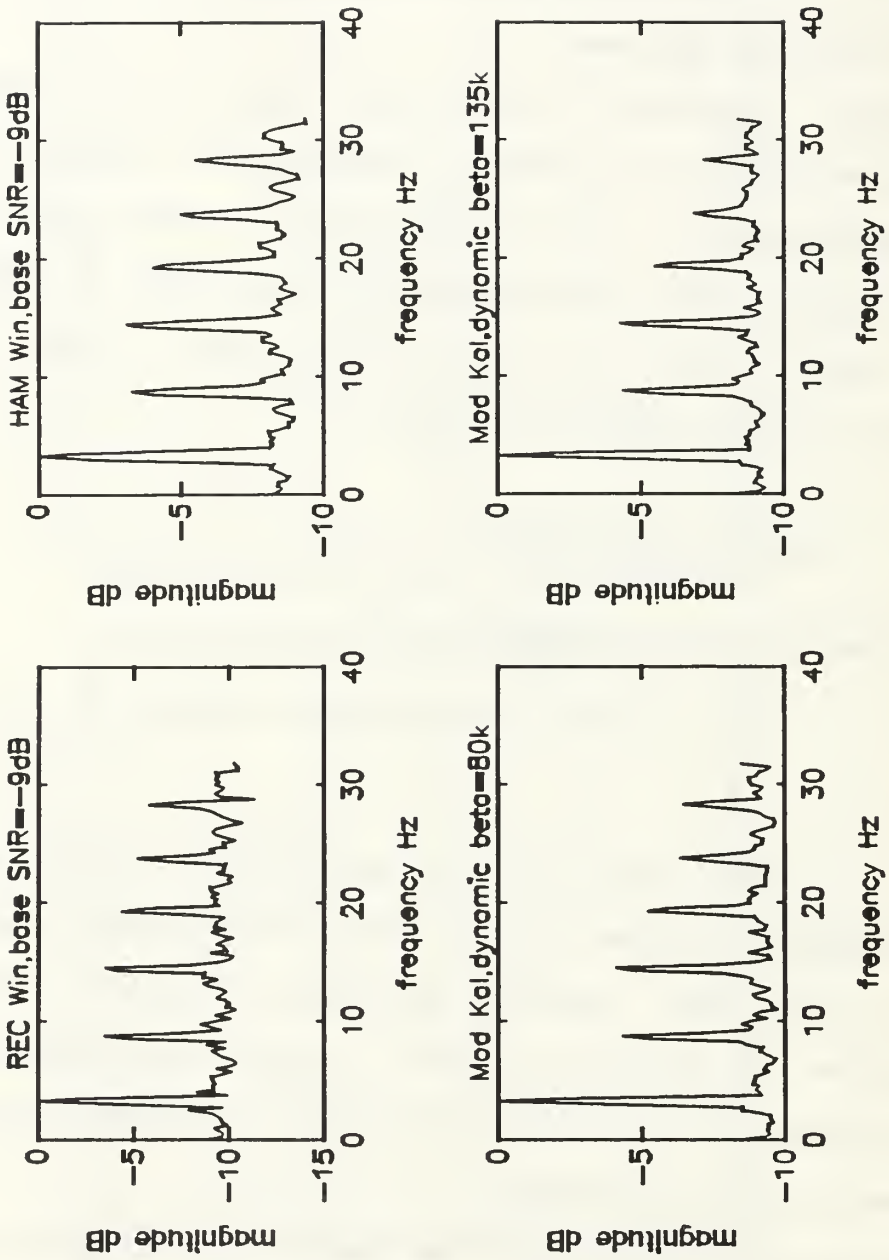

Figure 39. Averaging of LOFAR outputs using Hamming window and "modified" Kalman (dynamic β=80k and 135k). Freqs(Hz) at $SNRs_{in}$(dB); 3.3 at -9, 8.7 at -12, 14.4 at -13, 19.3 at -14, 23.7 at -15 and 28.3 at -16.

For completeness Figure 40 shows the LOFAR outputs for the Hamming window and the Kalman ("modified" dynamic β=135k) filters for this experiment. Neither process appears to have much success at detecting any of the six frequency components. Figure 41 is the output of the process defined in equation (25). The "modified" Kalman filter is much less successful at smoothing the noise variance between frequency components in this SNR range. This is due to the tendency of the Kalman filter to track the noise at small SNR's (also see Figure 37).

The tendency of the Kalman filter to track the noise at small SNR's is a result of the statistical properties of the FFT of additive white Gausian noise (WGN). If the noise input (time series) is a white Gaussian sequence with zero mean and variance σ^2, then the magnitude squared of the FFT $|Y(k)|^2$ will have a chi-squared distribution with 2 degrees of freedom with a variance σ^2_Y where the mean equals the standard deviation.

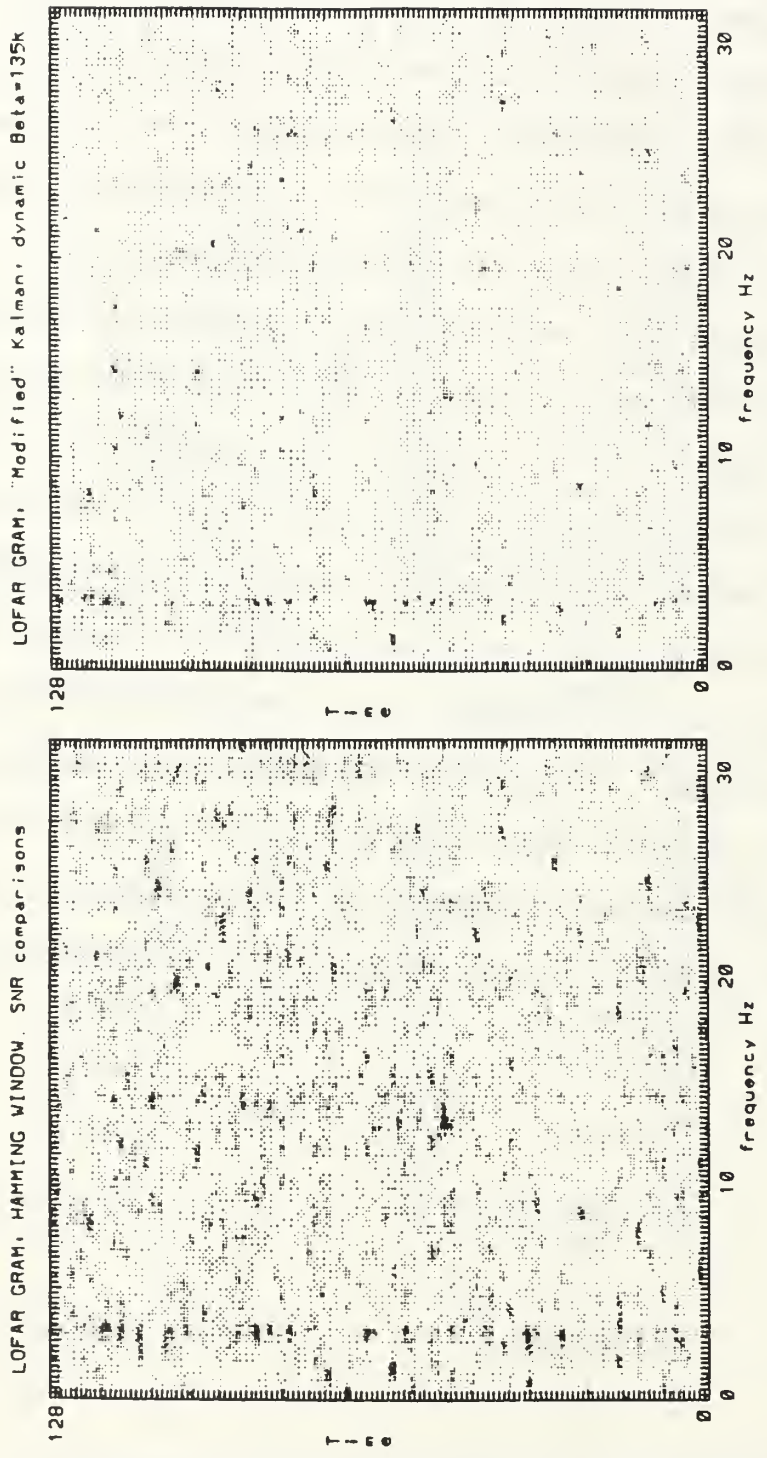

Figure 40. LOFAR outputs of Hamming window and "modified" Kalman (dyn β=135k) filter. Freqs at f1=3.3Hz, f2=8.7Hz, f3=14.4Hz, f4=19.3Hz, f5=23.7Hz and f6=28.3Hz with SNR_{in}'s of -16, -17, -18, -19, -20 and -21 dB respectively.

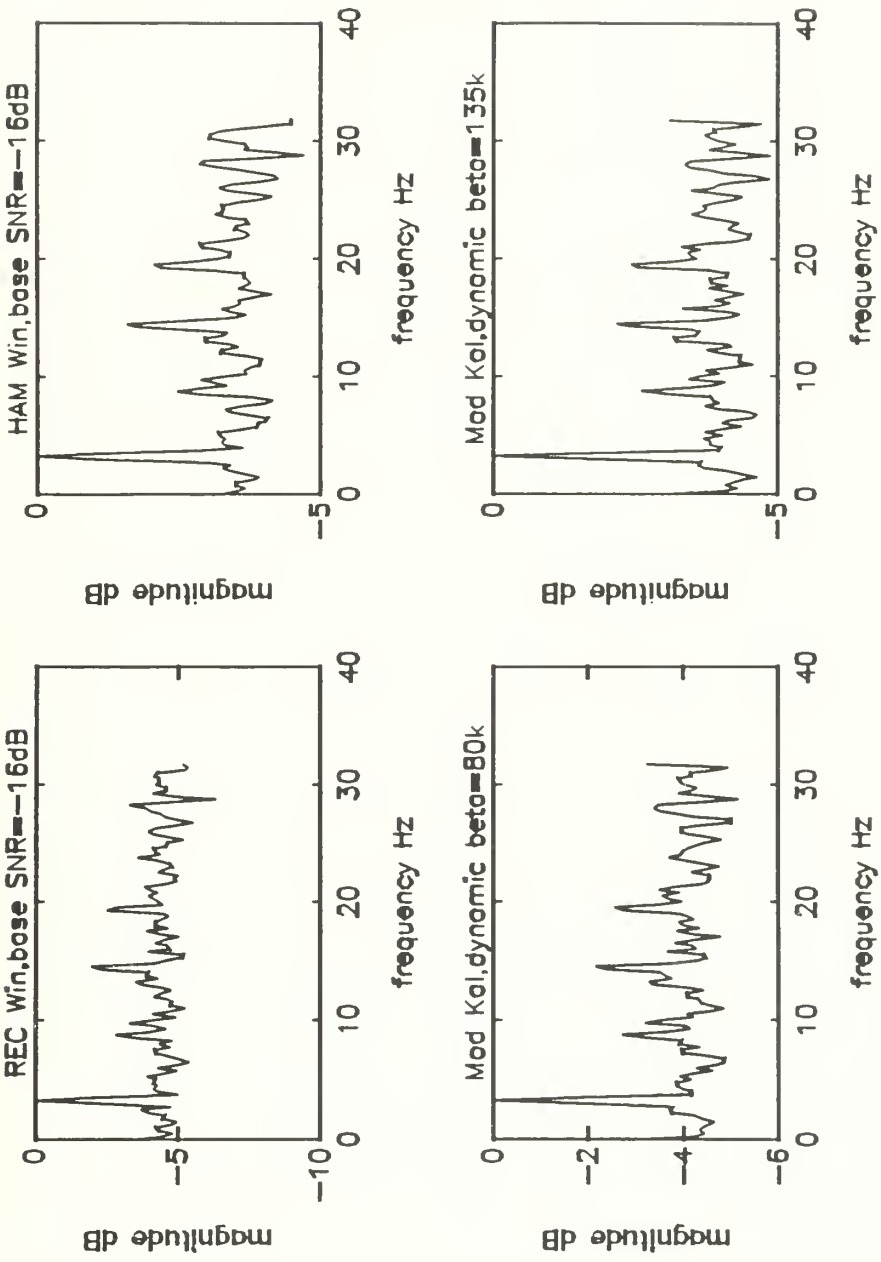

Figure 41. Averaged LOFAR outputs. Hamming window, "modified" Kalman using dynamic β of 80k and 135k. Freqs; f1=3.3Hz, f2=8.7Hz, f3=14.4Hz, f4=19.3Hz, f5=23.7Hz and f6=28.3Hz with SNR_{in} of -16, -17, -18, -19, -20 and -21 dB.

V. SUMMARY

A. CONCLUSIONS

The Kalman filters examined in this study (constant β, dynamic β and "modified" filter) all demonstrated the ability to enhance the spectral peaks of a periodogram of an unwindowed time series. The dynamic β and "modified" dynamic β filters provide the best resolution preservation of the mainlobe with negligible broadening. The Kalman filter significantly outperformed the Hamming window at $SNR's_{out}$ of 6 and 9 dB for β=135k and β=80k. The false detections for these filters cause negligible degradation to the output when displayed as LOFAR gram.

The selection of the filter parameter β is dependent on SNR_{out}. In addition, the selection of the filter parameter β can be adjusted to maximize filter performance if prior knowledge of the desired filter output is available. Using the "modified" filter with β=135k results in reliable signal detection at SNR_{out} of 9 dB in the periodogram with minimal false detections. For β=80k, reliable detections are also made at SNR_{out} of 9 dB but at the cost of an increase in false detections. Prior knowledge of the composition of the expected signal of interest will have a major impact on the selection of β. The larger the expected SNR_{out}, the larger β

can be to provide good smoothing with maximal signal detectability. For SNR_{out} of 6 dB, a β of 135k provids the best results. For SNR_{out} less than 6 dB, a β of 80k is the optimum choice while for SNR_{out} greater than 9 dB, a β of 300k is appropriate.

Viewing Kalman filter outputs on LOFAR outputs has resulted in detection of signals with SNR_{out} down to 3 dB. The LOFAR output also demonstrates the capability for the Kalman filter to process time varying spectra. See Appendix C for a more detailed description.

The "modified" Kalman filter (for either constant or dynamic β) can enhance the spectral peaks of a periodogram by consistently reducing the noise floor by 1 to 3 dB over the non-modified filter. This reduction in noise can have a dramatic impact when displayed as a LOFAR output.

B. FUTURE WORK

The Kalman filter is effective at preserving the spectral resolution of the original periodogram. Modifying the filter algorithm to substitute the original periodogram data for the filter estimates during up transitions enhances the Kalman filter performance in signal detection, noise reduction and spectral preservation. With the recent interest in signals with narrowband mainlobe structure (called "SWATH" signals), further modifications of the Kalman algorithm for specific

application to the processing of these "SWATH" components is an area of future work.

APPENDIX A. USERS GUIDE / COMPUTER CODE

A. SIMULATION USERS GUIDE

The Kalman filtering algorithms used in this study are written in Pro-Matlab (version 3.5h). All algorithms used to support the simulations are also written in Matlab with the exception of the grey-tone shading program which is written in FORTRAN 77. The simulations were run on a NPS Sun work station. The NPS Sun stations use the UNIX operating system via an Internet Networking System.

A complete simulation (from data generation to LOFAR output) can be executed providing all of the following code is copied into the directory in which Matlab is started in. This simulation is limited to a single stationary frequency component signal processed using the "modified" Kalman filter with a dynamic beta. The following steps should be followed to run this simulation:

1. Change to the directory that all applicable programs are residing in.
2. Start matlab by typing 'matlab1'.
3. At the matlab prompt type 'kfilter'.
4. Follow the instructions on the screen.

The entire simulation will take approximately 15 minutes for a 128 by 128 Kalman filtered array output. If a hard copy

output is desired follow the procedures provided in Appendix D. The program KFILTER.M can be modified to generate more complicated data for processing.

B. COMPUTER CODE

All the computer code needed to reproduce the work in this thesis is provided below.

1. Kalman Filter Algorithms

a. "Modified" Kalman with Dynamic β

```
%   KALMOD1.M
%   This program filters the array of desired periodograms
%   of time series data using a "modified" kalman filter with
%   a dynamic Beta. The entering arguments are;
%       - <file name> of PSD array
%       - sample size
%       - sigma (already calculated if data was generated
%         using "KFILTER.M" program).
%       - total number of (n•m) points in the array (already
%         calculated if "KFILTER.M" was used).
%       - the kalman filter parameter "Beta"
%   The subroutine outputs are;
%      M= periodogram of filter output in dB
%      C= periodogram of filter output in power

%   Define the function "KALMOD1"
    function [M,C]=kalmod1(psd,sampsize,sigma,totpts,betah)


%         KALMAN FILTER

          x=zeros(2,sampsize);
          pointer=zeros(2,sampsize);

     beta=betah;
     betal = 1
          sv2=sigma^2;

%   create loop for the number of rows in psd
    rows=totpts/sampsize;
    for k=1:rows

          y=psd(k,:);
```

```
        e1=0.0;
        e2=0.0;

        d1=0.0;
        d2=0.0;

        x(1,1)=y(1);
        x(2,1)=y(1);
        tau=1.0;

%       MAIN LOOP

        for t=1:(sampsize-1);
                k1=1.0/(tau+1.0);
                x11=x(1,t)+k1*(y(t)-x(1,t));
                d11=d1-beta;
                e11=e1+(1.0/(2.0*sv2))*((y(t+1)-x11)^2);
                c11=e11+d11;

                k2=1.0;
                x12=x(1,t)+k2*(y(t)-x(1,t));
                d12=d1+beta;
                e12=e1+(1.0/(2.0*sv2))*((y(t+1)-x12)^2);
                c12=e12+d12;

                k1=0.5;
                x21=x(2,t)+k1*(y(t)-x(2,t));
                d21=d2-beta;
                e21 = e2+(1.0/(2.0*sv2))*((y(t+1)-x21)^2);
                c21=e21+d21;

                k2=1.0;
                x22=x(2,t)+k2*(y(t)-x(2,t));
                d22=d2+beta;
                e22=e2+(1.0/(2.0*sv2))*((y(t+1)-x22)^2);
                c22=e22+d22;

%       UPDATE STATES IN DYNAMIC PROGRAM.

                  if c11<c21
                        x(1,t+1)=x11;
                        e1=e11 ;
                        d1=d11 ;
                        c1=e1+d1 ;
                        pointer(1,t+1)=1    ;
                        tau = tau+1;
                        beta=betah;
                   else
                        x(1,t+1)=y(t);
                        e1=e21;
                        d1=d21;
```

```
                        c1=e1+d1;
                        tau=2;
                        pointer(1,t+1)=2;
                        beta=beta1;
                    end

                    if c22<c12
                        x(2,t+1)=y(t);
                        e2=e22;
                        d2=d22;
                        c2=e2+d2;
                        pointer(2,t+1)=2;
                        beta=beta1;
                    else
                        x(2,t+1)=x12;
                        e2=e12;
                        d2=d12;
                        c2=e2+d2;
                        pointer(2,t+1)=1;
                        beta=betah;
                     end
            end

%           END MAIN LOOP

%           BACKWARDS SMOOTHING AND SUBSTITUTION

            tau=1.0;

            if c1<c2
                    out=1;
            else
                    out=2;
            end

            y(sampsize)=x(out,sampsize);

            for t=sampsize:-1:2
                      out=pointer(out,t);
                      xout=x(out,t-1);

                      if out==2
                              tau=1.0;
                              y(t-1)=xout;
                      else
                              tau=tau+1;

                              y(t-1)=xout;
                      end

                 trans(k,t-1)=out;
```

```
            C(k,:)=[ y(2:length(y))  y(1)];
        end
    end

%   Normalize each row of C to 1 and
%       clip the data to 10^(-6).

    psnorm=normod(C,rows);
    psnorm=clip(psnorm,10^(-6));

%   calculate the SPL of psnorm and store in M
    M=log10(psnorm).*10;
```

b. Kalman Filter using a Dynamic β

```
%   KALMOD.M
%   This program filters the array of desired periodograms
%   of the time series data using a kalman filter with a
%   dynamic Beta. The entering arguments are;
%       - <file name> of PSD array
%       - sample size
%       - sigma (already calculated if data was generated
%         using "KFILTER.M" program.
%       - total number of (n•m) points in the array (already
%         calculated if "KFILTER.M" was used.
%       - the kalman filter parameter "Beta"
%   The subroutine outputs are;
%       M= periodogram of filter output in dB
%       C= periodogram of filter output in power

%   Define the function "KALMOD"
    function [M,C]=kalmod(psd,sampsize,sigma,totpts,betah)

%       KALMAN FILTER

        x=zeros(2,sampsize);
        pointer=zeros(2,sampsize);

        sv2=sigma^2;
        beta=betah
        beta1=1

%   create loop for the number of rows in psd
    rows=totpts/sampsize;
    for k=1:rows

        y=psd(k,:);
        e1=0.0;
        e2=0.0;
```

```
d1=0.0;          d2=0.0;

x(1,1)=y(1);
x(2,1)=y(1);

tau=1.0;

%       MAIN LOOP

for t=1:(sampsize-1);
        k1=1.0/(tau+1.0);
        x11=x(1,t)+k1*(y(t)-x(1,t));
        d11=d1-beta;
        e11=e1+(1.0/(2.0*sv2))*((y(t+1)-x11)^2);
        c11=e11+d11;

        k2=1.0;
        x12=x(1,t)+k2*(y(t)-x(1,t));
        d12=d1+beta;
        e12=e1+(1.0/(2.0*sv2))*((y(t+1)-x12)^2);
        c12=e12+d12;

        k1=0.5;
        x21=x(2,t)+k1*(y(t)-x(2,t));
        d21=d2-beta;
        e21 = e2+(1.0/(2.0*sv2))*((y(t+1)-x21)^2);
        c21=e21+d21;

        k2=1.0;
        x22=x(2,t)+k2*(y(t)-x(2,t));
        d22=d2+beta;
        e22=e2+(1.0/(2.0*sv2))*((y(t+1)-x22)^2);
        c22=e22+d22;

%       UPDATE STATES IN DYNAMIC PROGRAM.

          if c11<c21
                x(1,t+1)=x11;
                e1=e11 ;
                d1=d11 ;
                c1=e1+d1 ;
                pointer(1,t+1)=1    ;
                tau = tau+1;
                beta=betah;
           else
                x(1,t+1)=x21;
                e1=e21;
                d1=d21;
                c1=e1+d1;
                tau=2;
                pointer(1,t+1)=2;
```

```
                    beta=betal;
                end

                if c22<c12
                      x(2,t+1)=x22;
                      e2=e22;
                      d2=d22;
                      c2=e2+d2;
                      pointer(2,t+1)=2;
                      beta=betal;
                else
                      x(2,t+1)=x12;
                      e2=e12;
                      d2=d12;
                      c2=e2+d2;
                      pointer(2,t+1)=1;
                      beta=betah;
                 end
        end

%       END MAIN LOOP

%       BACKWARDS SMOOTHING AND SUBSTITUTION

        tau=1.0;

        if c1<c2
                out=1;
        else
                out=2;
        end

        y(sampsize)=x(out,sampsize);

        for t=sampsize:-1:2
                  out=pointer(out,t);
                  xout=x(out,t-1);

                  if out==2
                           tau=1.0;
                           y(t-1)=xout;
                  else
                           tau=tau+1;
                           y(t-1)=xout;
                  end

             trans(k,t-1)=out;
             C(k,:)=[ y(2:length(y))  y(1)];
        end
    end
```

```
%   Normalize each row of C to 1 and
%       clip the data to 10^(-6).

    psnorm=normod(C,rows);
    psnorm=clip(psnorm,10^(-6));

%   calculate the SPL of psnorm and store in M
    M=log10(psnorm).*10;
```

c. "Modified" Kalman Filter using Constant β

```
%   KALMAN1.M
%   This program filters the array of desired periodograms
%   of the time series data using a "modified" kalman filter
%   with a constant Beta. The entering arguments are;
%       - <file name> of PSD array
%       - sample size
%       - sigma (already calculated if data was generated
%         using "KFILTER.M" program).
%       - total number of (n•m) points in the array (already
%         calculated if "KFILTER.M" was used).
%       - the kalman filter parameter "Beta"
%   The subroutine outputs are;
%       M= periodogram of filter output in dB
%       C= periodogram of filter output in power

%   Define the function "KALMAN1"
    function [M,C]=kalman1(psd,sampsize,sigma,totpts,beta)

%        KALMAN FILTER

         x=zeros(2,sampsize);
         pointer=zeros(2,sampsize);

         sv2=sigma^2;

%   create loop for the number of rows in psd
    rows=totpts/sampsize;
    for k=1:rows

         y=psd(k,:);
         e1=0.0;
         e2=0.0;

         d1=0.0;
         d2=0.0;

         x(1,1)=y(1);
         x(2,1)=y(1);
```

```
        tau=1.0;

%       MAIN LOOP

        for t=1:(sampsize-1);
                k1=1.0/(tau+1.0);
                x11=x(1,t)+k1*(y(t)-x(1,t));
                d11=d1-beta;
                e11=e1+(1.0/(2.0*sv2))*((y(t+1)-x11)^2);
                c11=e11+d11;

                k2=1.0;
                x12=x(1,t)+k2*(y(t)-x(1,t));
                d12=d1+beta;
                e12=e1+(1.0/(2.0*sv2))*((y(t+1)-x12)^2);
                c12=e12+d12;

                k1=0.5;
                x21=x(2,t)+k1*(y(t)-x(2,t));
                d21=d2-beta;
                e21 = e2+(1.0/(2.0*sv2))*((y(t+1)-x21)^2);
                c21=e21+d21;

                k2=1.0;
                x22=x(2,t)+k2*(y(t)-x(2,t));
                d22=d2+beta;
                e22=e2+(1.0/(2.0*sv2))*((y(t+1)-x22)^2);
                c22=e22+d22;

%       UPDATE STATES IN DYNAMIC PROGRAM.

                  if c11<c21
                        x(1,t+1)=x11;
                        e1=e11 ;
                        d1=d11 ;
                        c1=e1+d1 ;
                        pointer(1,t+1)=1   ;
                        tau = tau+1;
                   else
                        x(1,t+1)=y(t);
                        e1=e21;
                        d1=d21;
                        c1=e1+d1;
                        tau=2;
                        pointer(1,t+1)=2;
                   end

                   if c22<c12
                         x(2,t+1)=y(t);
                         e2=e22;
                         d2=d22;
```

```
                        c2=e2+d2;
                        pointer(2,t+1)=2;
                    else
                        x(2,t+1)=x12;
                        e2=e12;
                        d2=d12;
                        c2=e2+d2;
                        pointer(2,t+1)=1;
                    end
            end

%           END MAIN LOOP

%           BACKWARDS SMOOTHING AND SUBSTITUTION

            tau=1.0;

            if c1<c2
                    out=1;
            else
                    out=2;
            end

            y(sampsize)=x(out,sampsize);

            for t=sampsize:-1:2
                    out=pointer(out,t);
                    xout=x(out,t-1);

                    if out==2
                            tau=1.0;
                            y(t-1)=xout;
                    else
                            tau=tau+1;
                            y(t-1)=xout;
                    end

                trans(k,t-1)=out;
                C(k,:)=[ y(2:length(y))  y(1)];
            end
        end

%       Normalize each row of C to 1 and
%           clip the data to 10^(-6).

        psnorm=normod(C,rows);
        psnorm=clip(psnorm,10^(-6));

%       calculate the SPL of psnorm and store in M
        M=log10(psnorm).*10;
```

d. Kalman Filter using a Constant β

```
%   KALMAN.M
%   This program filter the array of desired periodograms
%   of the time series data using a kalman filter with a
%   constant Beta. The entering arguments are;
%       - <file name> of PSD array
%       - sample size
%       - sigma (already calculated if data was generated
%         using "KFILTER.M" program).
%       - total number of (n•m) points in the array (already
%         calculated if "KFILTER.M" was used).
%       - the kalman filter parameter "Beta"
%   The subroutine outputs are;
%      M= periodogram of filter output in dB
%      C= periodogram of filter output in power

%   Define the function "KALMAN"
    function [M,C]=kalman(psd,sampsize,sigma,totpts,beta)

%        KALMAN FILTER

         x=zeros(2,sampsize);
         pointer=zeros(2,sampsize);

         sv2=sigma^2;

%   create loop for the number of rows in psd
    rows=totpts/sampsize;
    for k=1:rows

         y=psd(k,:);
         e1=0.0;
         e2=0.0;

         d1=0.0;
         d2=0.0;

         x(1,1)=y(1);
         x(2,1)=y(1);
         tau=1.0;

%        MAIN LOOP

         for t=1:(sampsize-1);
                 k1=1.0/(tau+1.0);
                 x11=x(1,t)+k1*(y(t)-x(1,t));
                 d11=d1-beta;
                 e11=e1+(1.0/(2.0*sv2))*((y(t+1)-x11)^2);
                 c11=e11+d11;
```

```
                k2=1.0;
                x12=x(1,t)+k2*(y(t)-x(1,t));
                d12=d1+beta;
                e12=e1+(1.0/(2.0*sv2))*((y(t+1)-x12)^2);
                c12=e12+d12;

                k1=0.5;
                x21=x(2,t)+k1*(y(t)-x(2,t));
                d21=d2-beta;
                e21 = e2+(1.0/(2.0*sv2))*((y(t+1)-x21)^2);
                c21=e21+d21;

                k2=1.0;
                x22=x(2,t)+k2*(y(t)-x(2,t));
                d22=d2+beta;
                e22=e2+(1.0/(2.0*sv2))*((y(t+1)-x22)^2);
                c22=e22+d22;

%       UPDATE STATES IN DYNAMIC PROGRAM.

                  if c11<c21
                        x(1,t+1)=x11;
                        e1=e11 ;
                        d1=d11 ;
                        c1=e1+d1 ;
                        pointer(1,t+1)=1   ;
                        tau = tau+1;
                   else
                        x(1,t+1)=x21;
                        e1=e21;
                        d1=d21;
                        c1=e1+d1;
                        tau=2;
                        pointer(1,t+1)=2;
                   end

                   if c22<c12
                         x(2,t+1)=x22;
                         e2=e22;
                         d2=d22;
                         c2=e2+d2;
                         pointer(2,t+1)=2;
                   else
                         x(2,t+1)=x12;
                         e2=e12;
                         d2=d12;
                         c2=e2+d2;
                         pointer(2,t+1)=1;
                    end
      end
```

```
%         END MAIN LOOP

%         BACKWARDS SMOOTHING AND SUBSTITUTION

          tau=1.0;

          if c1<c2
                  out=1;
          else
                  out=2;
          end

          y(sampsize)=x(out,sampsize);

          for t=sampsize:-1:2
                    out=pointer(out,t);
                    xout=x(out,t-1);

                    if out==2
                             tau=1.0;
                             y(t-1)=xout;
                    else
                             tau=tau+1;
                             y(t-1)=xout;
                    end

               trans(k,t-1)=out;
               C(k,:)=[ y(2:length(y))  y(1)];
          end
     end

%    Normalize each row of C to 1 and
%        clip the data to 10^(-6).

     psnorm=normod(C,rows);
     psnorm=clip(psnorm,10^(-6));

%    calculate the SPL of psnorm and store in M
     M=log10(psnorm).*10;
```

2. Supporting Matlab Code

a. Signal Generator

```
%    KFILTER.M            R.C. Adamo
%    The purpose of this program is to connect all ap-
%    propriate subroutines allowing the user to generate
%    a single stationary frequency component test signal
%    for processing.  The user is prompted for;
%        1) Input frequency
```

```
%       2) Sampling frequency
%       3) Sample size (Power of 2 please)
%       4) Signal length (Power of 2 please)
%       5) Input SNR
%    The signal is processed in non-overlapping
%    segments zero-padded to twice the sample size.
%    The following subroutines are called;
%       1) PER.M
%       2) KALMOD1.M
%          (KALMAN.M, KALMAN1.M AND KALMOD.M can also
%          be used if desired.  User must delete the
%          "%" preceding the desired subroutine call-up
%                in this program)
%    After processing, the square of the magnitudes of the
%    FFT's are saved in arrays of (signal length/sample
%    size) rows by sample size columns as;
%       - rpsd.dat
%       - hwpsd.dat
%       - kpsd_2B_mod.dat (if you modify this program to
%         run the other kalman algorithms you must save
%         that data manually)
%    For a signal length of (128*128), allow 10 mins per Kalman
%    filter utilized for total run time.

     clear
%    Input desired frequency
     f1=input('Enter Signal freq less than 256 Hz ')

%    Input sampling frequency
     fs=input('Enter sampling freq  ')

%    Input sample size
     size=input('Enter sample size (power of 2 pls) ')

%    Input the number of points to be generated.
     totpts=input('Enter total # of pts (power of 2 pls)')
     no=0:1:(totpts-1);

%    Input the desired SNR of this signal
     snr=input('Enter SNR in dB ')
     sigma=sqrt((10^(-snr/10))*.5)

%    Generate sinusoid only
     theta1=2*pi*f1*no;
     y1=sin(theta1/fs);
     ycomp=y1;

%    Add Gaussian noise
     rand('normal')
     rand('seed',5)
     ydirt=ycomp+sigma*rand(ycomp);
```

```
%   define beta parameters
    b1=input('Enter desired filtering beta ')

%   Call subroutines to compute rec,rpsd,h,hwpsd,kw,kwpsd
    [rec,rpsd,h,hwpsd,freq]=per(ydirt,size,fs);
%   [k_1B,kpsd_1B]=kalman(rpsd,size,sigma,totpts,b1);
%   [k_1B_mod,kpsd_1B_mod]=kalman1(rpsd,size,sigma,totpts,b1);
%   [k_2B,kpsd_2B]=kalmod(rpsd,size,sigma,totpts,b1);
    [k_2B_mod,kpsd_2B_mod]=kalmod1(rpsd,size,sigma,totpts,b1);

    save
    save rpsd.dat rpsd /ascii
    save hwpsd.dat hwpsd /ascii
    save kpsd_2B_mod.dat kpsd_2B_mod /ascii
quit
```

b. Subroutine "PER.M"

```
%   PER.M                          R.C. Adamo
%   Compute the rectangular window and the hamming window
%   periodograms and the frequency vector for the desired
%   signal.  The input arguments are signal name, the
%   sample size of the transforms and the sampling frequency.
%   The data segments are zero-padded to twice the sample
%   size.
%   The output data of the subroutine are:
%       M=periodogram of unwindowed time series in dB
%       B=periodogram of unwindowed time series in power
%       C=periodogram of hamming windowed time series
%         in dB
%       D=periodogram of hamming windowed time series
%         in power
%       E=sampling frequency vector used for plotting
%         individual periodograms.

%   Define the function "PER"
    function [M,B,C,D,E]=per(signal,sampsize,sampfreq)

    rows=length(signal)/sampsize;
    for k=1:rows

%   zero pad sample size pt data sets to twice the sample
%   size.
    temp(1:sampsize)=signal((k-1)*sampsize+1:k*sampsize);
    temp(sampsize+1:2*sampsize)=zeros(1:sampsize);

%   Square the magnitude of the fft of the padded sequence
    ztemp=(abs(fft(temp))).^2;

%   Store the left half of ztemp as Matrix where
```

```
%   Matrix is n by sample size matrix
    B(k,:)=ztemp(1:sampsize);

    clear temp ztemp
    end

%   Normalize each row of B and call it psnorm
    psnorm=normod(B,rows);

%   Clip psnorm to minimum value of 10^-6
    psnorm=clip(psnorm,10^(-6));


%   calculate the SPL of psnorm and store in M
    M=log10(psnorm).*10;

    [C,D]=hamm(signal,sampsize);
    E=fvec(sampfreq,sampsize);
```

c. Hamming Window Subroutine

```
%   HAMM.M                          R.C. Adamo
%   Compute the periodogram of desired signal after applying
%   a Hamming window to each time series data sample of
%   defined sample size. The input arguments are the signal
%   name and the transform sample size. The data segments are
%   zero-padded to twice the sample size.
%   The output data of the subroutine are:
%       M=periodogram of hamming windowed time series
%         in dB
%       B=periodogram of hamming windowed time series
%         in power

%   Define the function "HAMM"
    function [M,B]=hamm(signal,sampsize)

%   Create hamming window of in-putted sample size
    w=hamming(sampsize);

    rows=length(signal)/sampsize;
    for k=1:rows

%   apply hamming window and zero pad data to twice the sample
%   size.
    temp(1:sampsize)=w'.*signal((k-1)*sampsize+1:k*sampsize);
    temp(sampsize+1:2*sampsize)=zeros(1:sampsize);

%   Square the fft magnitude of the padded sequence
    ztemp=(abs(fft(temp))).^2;
```

```
%   Store the left half of ztemp as Matrix where Matrix
%   is n by sample size;  n=totpts/sample size.
    Matrix(k,:)=ztemp(1:sampsize);

    clear temp ztemp
    end

%   Normalize each row of Matrix and call it psnorm
    psnorm=normod(Matrix,rows);

    B=Matrix;

%   Clip psnorm to minimum value of 10^-6
    psnorm=clip(psnorm,10^(-6));

%   Calculate the SPL of psnorm and store in M
    M=log10(psnorm).*10;
```

d. Subroutine "NORMOD.M"

```
%   NORMOD.M                  R. C. Adamo
%   This program normalizes a matrix such
%   that the largest value in each row is
%   1 and the other values are scaled relative to 1.
%   The entering arguments are the matrix name and
%   the number of rows in the matrix.

%   Define the function "NORMOD"
    function M=normod(matrix,rows)

    for k=1:rows
       M(k,:)=(1/max(matrix(k,:))) * matrix(k,:);
    end
```

e. Subroutine "CLIP.M"

```
%   CLIP.M                    R.C.ADAMO
%   This subroutine clips any matrix to some
%   minimum value.  The entering arguments are the
%   matrix to be clipped and the minimum value the user
%   wishes to clip it to.

%   Define the function "CLIP"
    function M=clip(matrix,clip)

    index = matrix < (clip);
    corr = index * clip;
    corfac = 1 - index;
    adjust = corfac.*matrix;
    M = corr + adjust;
```

f. Subroutine "NORMAL.M"

```
%   NORMAL.M                    R. C. Adamo
%   This program normalizes a matrix such
%   that the largest value in the matrix are
%   1 and the other values are scaled relative to 1.

%   Define the function "NORMAL"
    function M=normal(matrix)

    M=(1/max(max(matrix)))*matrix;
```

g. Subroutine "FVEC.M"

```
%   FVEC.M                        R.C. Adamo
%   CREATE THE FREQUENCY VECTOR USED IN PLOTTING
%   A PERIODOGRAM. fs IS THE SAMPLING FREQUENCY
%   AND X IS THE SAMPLE SIZE.

        function f=fvec(fs,x)

        n=2*x;
        f=fs*(0:n-1)/n;
```

h. Grey-tone imaging FORTRAN 77 Code

```
C   LOFAR.F                             R.C.ADAMO
C   This program reads and displays processed signals
C   in a LOFAR presentation.  It is used to compare
C   the performances of different processing techniques.
C   The user must edit this program to ensure input filedef
C   is properly defined. Also this program must be edited to
C   add a proper title on output.
C
C OPEN GKS, OPEN WORKSTATION OF TYPE 1, ACTIVATE WORKSTATION
C
      CALL GOPKS (6,IDUM)
      CALL GOPWK (1, 2, 1)
      CALL GACWK (1)
C
C INVOKE DEMO DRIVER
C
      call thafto(ierr)
C
C     DEACTIVATE AND CLOSE WORKSTATION, CLOSE GKS.
C
      CALL GDAWK (1)
      CALL GCLWK (1)
      CALL GCLKS
      STOP
```

```
      END

      SUBROUTINE THAFTO (IERROR)
C
C USAGE                    CALL THAFTO (IERROR)
C
C ARGUMENTS
C ON OUTPUT                IERROR
C                            An integer variable
C                            = 0, if the test was successful,
C                            = 1, the test was not successful.
C
C I/O          If the test is successful, the message
C              HAFTON EXECUTED--SEE PLOTS TO CERTIFY
C             is printed on unit 6.  In addition, 1 half-tone
C              frame is produced on the machine graphics
C              device.  In order to determine if the test
C              was successful, it is necessary to examine
C              the plot.
C
C PRECISION                Single
C
C LANGUAGE                 FORTRAN 77
C
C REQUIRED ROUTINES        HAFTON
C
C REQUIRED GKS LEVEL       0A
C
C Z contains the values to be plotted.
C
      REAL             Z(128,128)
      INTEGER          ROWS, SIZE
      SAVE
C
C Specify coordinates for plot titles.  The values TX and TY
C define the center of the title string in a 0. to 1. range.
C
      DATA TX/0.15/, TY/0.9469/
C
C Specify low (FLO) and high (FHI) contour values, and NLEV
C unique contour levels.  NOPT determines how Z maps onto the
C intensities, and the directness of the mapping.
C
      DATA FLO/0.0/, FHI/0.0/, NLEV/1/, NOPT/1/
C
C Initialize the error indicator
C
      IERROR = 0
C
C Define files to be read in.
    open(1,file='rpsd.dat')
```

```
C    open(1,file='h_snrcomp1.dat')
C    open(1,file='k_2B_mod_135k.dat')
C
C    Input data array dimensions
       print *, ' Enter array size as rows, column '
       read *, rows, size
C       rows=128
C       size=128

C Read in the desired data to be plotted
C
       DO  20 I=1,rows
          READ(1,*,END=999) (Z(J,I),J=1,size)
    20 CONTINUE
C
C Select normalization trans 0 for plotting title.
C
       CALL GSELNT (0)
C
C Call WTSTR to write the plot title.
C
C Plot upper title for output
       CALL WTSTR (TX,TY,
      1           'LOFAR GRAM; RECT WINDOW; Base SNR out=9
      1 dB',2,0,-1)
C      1           'LOFAR GRAM; HAMM WINDOW; Base SNR out=9
C      1  dB',2,0,-1)
C      1          'LOFAR GRAM; "Mod" Kalman dynamic Beta=135k'
C      1 ,2,0,-1)
C
C Print frequency scale on output
       CALL WTSTR (0.10,0.08,
      1 '0               10               20               30
C     1 ',2,0,-1)
C
       CALL WTSTR (0.40,0.055,
      1          'frequency Hz',2,0,-1)
C
C Entry HAFTON allows user specification of plot parameters.
C
       CALL HAFTON (Z,size,size,rows,FLO,FHI,NLEV,NOPT,0,0,0.)
       CALL FRAME
C
       WRITE (6,1001)
       RETURN
 1001 FORMAT (' HAFTON TEST EXECUTED--SEE PLOTS TO CERTIFY')
C
  999 write(6,1111)
 1111 format('unexpected end of file on input: program
C     1 terminated')
       END
```

i. Subroutine "NAMES.M"

```
%   NAMES.M                    R.C. Adamo
%   Create the titles used for the comparison plots

    dBname=[sprintf('SNR=%g',snr),'dB,']
    Betaname=sprintf('BETA=%g',b1)
    freqname=[sprintf('freq=%g',f1),'Hz']
    Rectitle=['REC Win,',dBname,freqname]
    HWtitle=['HAM Win,',dBname,freqname]
    Kaltitle=['Mod Kal,dynamic beta, ',Betaname]
    K1title =['Mod Kal, constant beta, ',Betaname]
    K2title =['Kal Out,dynamic beta, ',Betaname]
    K3title =['Kal Out, constant beta ,',Betaname]
    Betahigh=sprintf('Beta high = %g',b1)
    Betalow= sprintf('Beta low = %g',1)
```

j. Plotting Subroutine

```
%   PLOTT.M                          R.C. Adamo
%   This subroutine plots and compares the periodograms
%   of the rectangular and Hamming windows and the Kalman
%   filter outputs in dB. If other than the default
%   kalman algorithm is used, the user must modify line 18
%   of this program to plot the desired kalman output.
%   This program must be run after executing the program
%   "names" in matlab.

    for i=1:1:totpts/size
clg
        subplot(221)
    plot(freq(1:size),rec(i,:))
    title(Rectitle)
    xlabel('frequency Hz')
    ylabel('magnitude dB')
    plot(freq(1:size),h(i,:))
    title(HWtitle)
    xlabel('frequency Hz')
    ylabel('magnitude dB')
    plot(freq(1:size),k_2B_mod(i,:))
    title(Kaltitle)
    xlabel('frequency Hz')
    ylabel('magnitude dB')
    text(.60,.30,Betahigh,'sc')
    text(.60,.25,Betalow,'sc')
    text(.60,.20,sprintf('Noise realization %g',i),'sc')
pause
    end
```

APPENDIX B. PERFORMANCE RELATIVE TO SPECTRAL RESOLUTION

The ability of the Kalman filter to detect two components at nearby spectral locations was examined. Three test cases were evaluated to determine the minimum spectral separation needed by the Kalman filter to distinguish between two components. The same data sets were also processed using the Hamming window for comparison.

All three test cases consisted of three sinusoidal components in additive Gaussian noise. The frequencies in test case 1 were 10.7 Hz, 11.2 Hz and 20.1 Hz at input SNR_{in} of -9 dB, -12 dB and -15 dB, respectively. This case provides a 2 bin separation between f1 (10.7 Hz) and f2 (11.2 Hz). This signal was processed using the Hamming window and the Kalman ("modified" dynamic β=135k) filter. Test frequencies in test case 2 were 10.7 Hz, 11.45 Hz and 20.1 Hz at input SNR_{in} of -9 dB, -12 dB and -15 dB (3 bin separation between f1 and f2), respectively. Finally, test frequencies in test case 3 were 10.7 Hz, 11.7 Hz and 20.1 Hz at input SNR_{in} of -9 dB, -12 dB and -15 dB (4 bin separation between f1 and f2), respectively. Figure 42 thru Figure 44 shows the LOFAR outputs for the Hamming window and the Kalman ("modified" dynamic β=135k) for cases 1, 2 and 3, respectively. The mainlobe broadening of the Hamming window over that of the Kalman filter is obvious

in all three cases. In addition, the Kalman filter clearly detects the distinct frequencies of f1 and f2 in case 3 (4 bin separation) where the Hamming window process can not. It is difficult to pull out f3, the -15 dB SNR_{in} component, using either the Hamming or Kalman filter in any of the three cases.

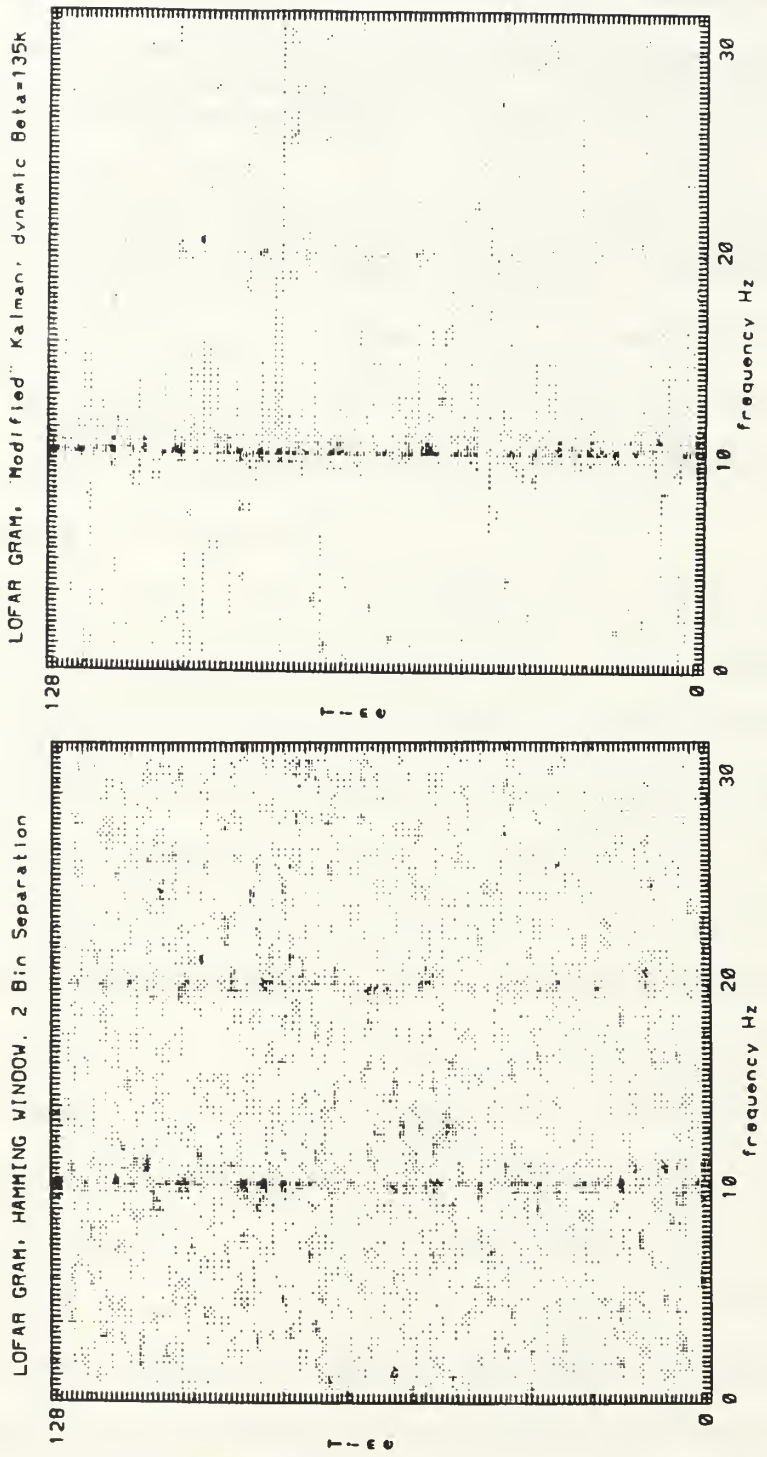

Figure 42. 2 BIN SEPARATION. Sinusoid (f1=10.7Hz at SNR_{in}= -9dB, f2=11.2Hz at SNR_{in}=-12dB and f3=20.1Hz at SNR_{in}=-15dB) comparing Hamming vs Kalman ("modified" dynamic β=135k).

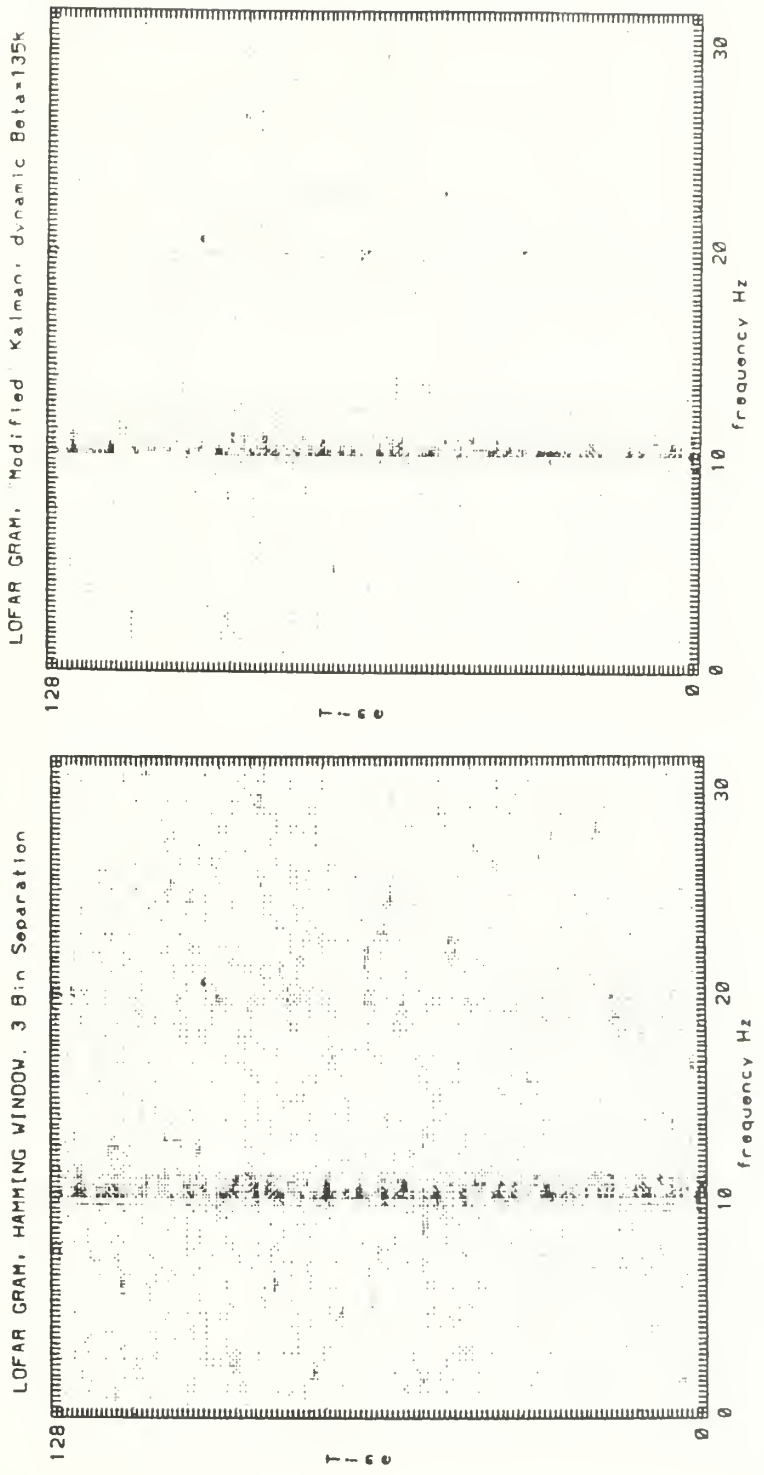

Figure 43. 3 BIN SEPARATION. Sinusoid (f1=10.7Hz at SNR_{in}=-9dB, f2=11.45Hz at SNR_{in}=-12dB and f3=20.1Hz at SNR_{in}=-15dB) comparing Hamming vs Kalman ("modified" dynamic β=135k).

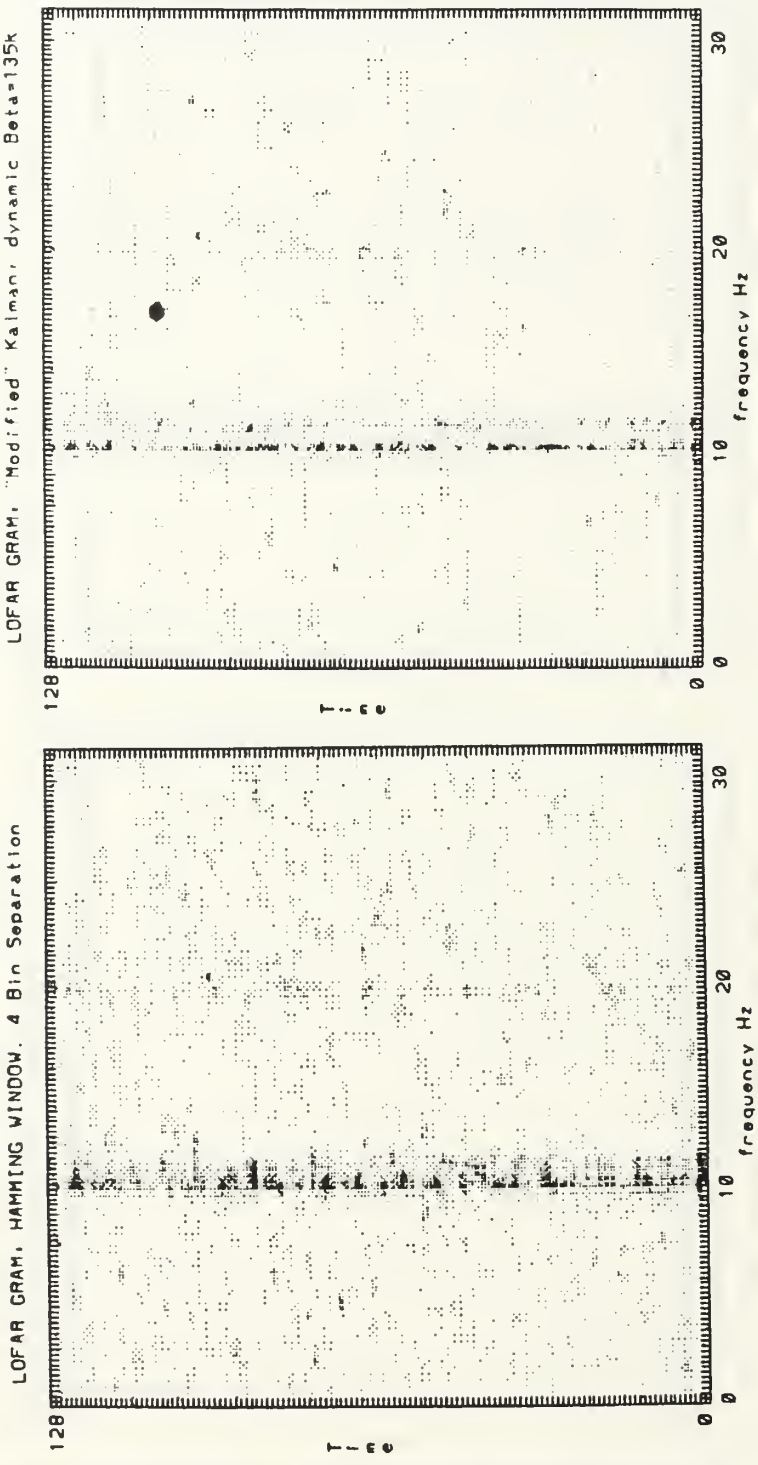

Figure 44. 4 BIN SEPARATION. Sinusoid (f1=10.7Hz at SNR_{in}=-9dB, f2=11.7Hz at SNR_{in}=-12dB and f3=20.1Hz at SNR_{in}=-15dB) comparing Hamming vs Kalman ("modified" dynamic β=135k).

APPENDIX C. EFFECTS OF DYNAMIC SPECTRAL COMPONENTS

LOFAR outputs are extremely useful in providing the additional processing gain needed to detect signal in noise at SNR_{out} of 6 dB or less. Up to this point the experimental data was comprised of stationary frequency components. It has been demonstrated that the human eye is well suited to linearly regress the data displayed via the LOFAR output for stationary frequency components. The effects that dynamic spectral components will have on this capability is examined in this appendix. The time series data for this experiment was generated with three frequency components. One component is a stationary frequency of 20.45 Hz at an SNR_{in} of -15 dB. The second spectral component is a linear ramp frequency with an SNR_{in} of -12 dB starting at 11.45 Hz and ending at 31.45 Hz. The third component is a dynamic component with an SNR_{in} of -9 dB starting and ending at 15.7 Hz and completing one sine wave cycle of amplitude of 2.5 Hz.

Figure 45 is the LOFAR output of this data filtered by the Hamming window and the Kalman ("modified" dynamic β=135k) filter. Both dynamic components are clearly visible in both the Hamming and Kalman LOFAR outputs. LOFAR output allows the humman eye to observe the structure of dynamic spectral components at $SNR's_{out}$ down to low as 6 dB.

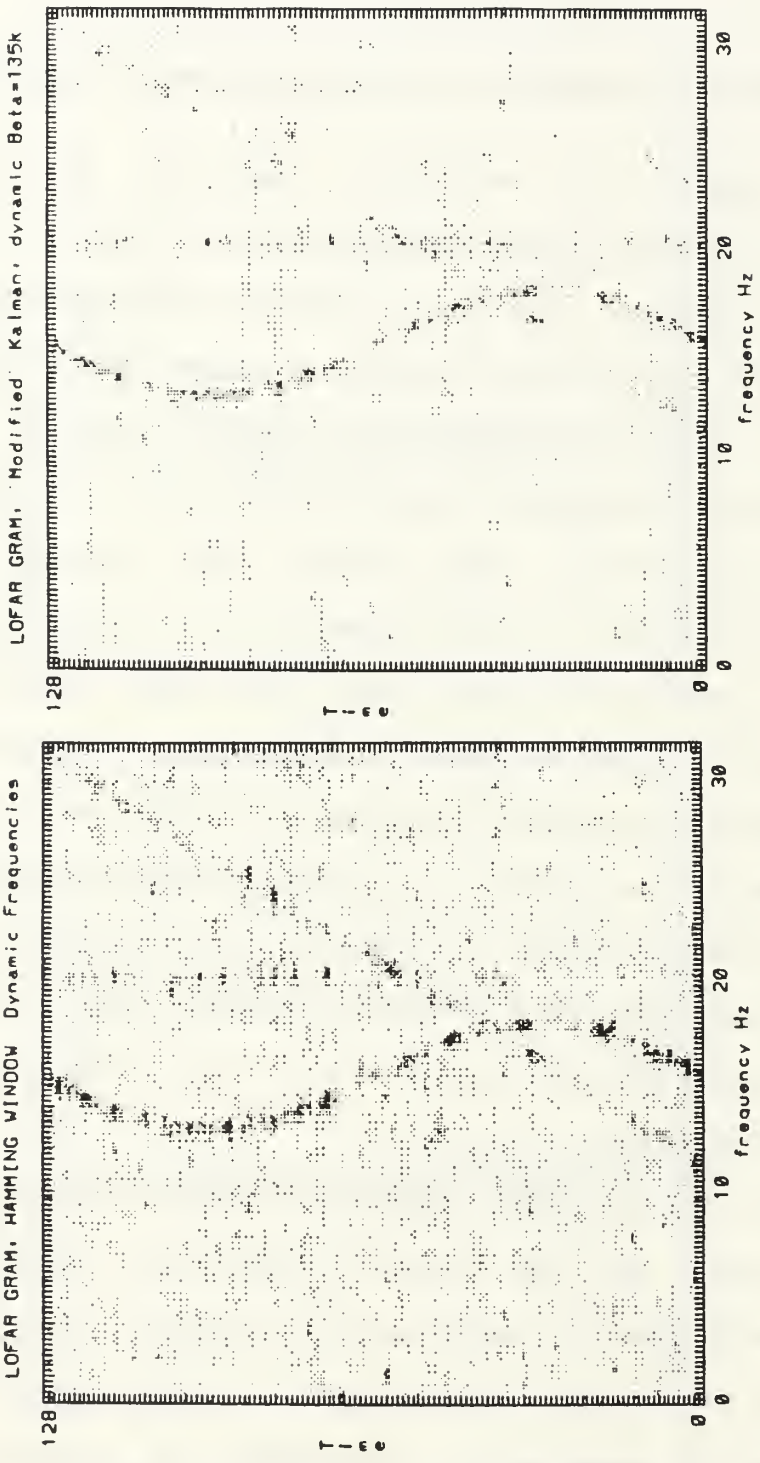

Figure 45. LOFAR comparison of Hamming window vs Kalman ("modified" DYN β=135k) of dynamic freqs (sine wave at SNR_{in}=-9 dB, ramp at SNR_{in}=-12 dB,fixed at SNR_{in}=-15 dB).

APPENDIX D. LOFAR OUTPUT USERS GUIDE

A. USERS GUIDE

This appendix provides the computer code and the step-by-step instructions necessary to produce a LOFAR output of any array of data. The user is required to load the data of interest into the same directory that the LOFAR code "image.f" is using. The data file must be named "input.dat". When the data is in place the following procedures should be followed:

i) For output to be displayed on the Sun workstation:

1. Rename data file to "input.dat".
2. At the prompt type "ncargf77 -o image image.f". This compiles the program and names the executable file "image".
3. At the prompt type "ncargrun -o image.cgm image". This executes image and saves the LOFAR output as "image.cgm"
4. The program prompts the user to input the number of rows and number of columns in their data. At the prompt type "(# of rows) <space> (# of columns)". The program then reads in the data and builds the LOFAR output.
5. At the prompt type "ctrans -d sunview image.cgm". This translates the "image.cgm" to the appropriate form for display on the Sun workstation.

ii) Hard copy LOFAR output

Once the procedures described above have been executed, hard copy of the LOFAR output ("image.cgm") can be produced on

the tektronix 4014 (tek4014) terminal in the computer lab Sp-301 if desired. To obtain a hard copy output follow these procedures:

1. Using the tek4014 terminal in Sp-301 log on to the VAX and type "tek4014" at the prompt <term(vt200)>.

2. Remote logon to the DSP used to produce the LOFAR output on. Again type "tek4014" at the <term(vt200)> prompt.

3. Change to the directory where the LOFAR output file "image.cgm" resides.

4. Type "ctrans -d t4010 image.cgm".

5. It takes between 1-2 minutes to display the LOFAR output on the tektronix. Once plotting is complete a hard copy can be obtained by pressing the "copy button" on the Tektronix 4631 Hard Copy Unit.

B. COMPUTER CODE

```
C   IMAGE.F                    R.C.ADAMO
C   This program reads and displays data sets of
C   "n by m" dimensions.  The user must ensure the data
C   is saved under the name of "INPUT.DAT". The user is
C   prompted for the dimensions of the data set. The
C   number of rows and columns must be integers.
C
C OPEN GKS, OPEN WORKSTATION OF TYPE 1, ACTIVATE WORKSTATION
C
      CALL GOPKS (6,IDUM)
      CALL GOPWK (1, 2, 1)
      CALL GACWK (1)
C
C INVOKE DEMO DRIVER
C
      call thafto(ierr)
C
C     DEACTIVATE AND CLOSE WORKSTATION, CLOSE GKS.
C
      CALL GDAWK (1)
      CALL GCLWK (1)
      CALL GCLKS
      STOP
      END
```

```
      SUBROUTINE THAFTO (IERROR)
C
C
C
C USAGE                    CALL THAFTO (IERROR)
C
C ARGUMENTS
C ON OUTPUT                IERROR
C                            An integer variable
C                            = 0, if the test was successful,
C                            = 1, the test was not successful.
C
C I/O        If the test is successful, the message
C            HAFTON EXECUTED--SEE PLOTS TO CERTIFY
C
C            is printed on unit 6.  In addition, 1 half-tone
C            frame is produced on the machine graphics
C            device.  In order to determine if the test
C            was successful, it is necessary to examine
C            the plot.
C
C PRECISION                Single
C
C LANGUAGE                 FORTRAN 77
C
C REQUIRED ROUTINES        HAFTON
C
C REQUIRED GKS LEVEL       0A
C
C
C Z contains the values to be plotted.
C
      REAL             Z(128,128)
      INTEGER          ROWS, SIZE
      SAVE
C
C Specify coordinates for plot titles.  The values TX and TY
C define the center of the title string in a 0. to 1. range.
C
      DATA TX/0.15/, TY/0.9469/
C
C Specify low (FLO) and high (FHI) contour values, and NLEV
C unique contour levels.  NOPT determines how Z maps onto the
C intensities, and the directness of the mapping. Execute
C "man HAFTON" for a detailed description of these variables.
C
      DATA FLO/0.0/, FHI/0.0/, NLEV/1/, NOPT/1/
C
C Initialize the error indicator
C
      IERROR = 0
```

```
C
C Define file to be read in.
    open(1,file='input.dat')
C
C
C Is data file called INPUT.DAT?
       print *, ' Is your data file named INPUT.DAT?'
       print *, '     Enter 1 for yes and '
       print *, '           2 for no.'
       print *, '     '
       print *, '  If you answer no the program will '
       print *, '  terminate. Rename file and try again.'
       read *, k
       if (k.eq.2) go to 991

C   Input data array demensions
       print *, ' Enter array size as rows, column '
       read *, rows, size

C Read in the desired data to be plotted
C
      DO  20 I=1,rows
         READ(1,*,END=999) (Z(J,I),J=1,size)
   20 CONTINUE
C
C Select normalization trans 0 for plotting title.
C
      CALL GSELNT (0)
C
C
C
C Call WTSTR to write the plot title.
C
      CALL WTSTR (TX,TY,
     1          'Your Plot tilte goes here',2,0,-1)
      CALL WTSTR (0.15,0.08,
     1         'Scale data could go here',2,0,-1)
C
C
C Entry HAFTON allows user specification of plot parameters.
C
      CALL HAFTON (Z,size,size,rows,FLO,FHI,NLEV,NOPT,0,0,0.)
      CALL FRAME
C
      WRITE (6,1001)
      RETURN
C
 1001 FORMAT (' HAFTON TEST EXECUTED--SEE PLOTS TO CERTIFY')
C
  999 write(6,1111)
 1111 format('unexpected end of file on input: program
```

```
    1  terminated')
     go to 9999
 991 print *, 'rename input file to INPUT.DAT and try again'
9999 END
```

LIST OF REFERENCES

1. House W.C., *Interactive Computer Graphics Systems*, p. 5, Petroceli Books, Inc., 1982.

2. Sanders M.S. and McCormick E.J., *Human Factors in Engineering Design*, 6th ed., pp. 46-47, McGraw-Hill, 1987.

3. Woodson, W.E., *Human Factors Design Handbook*, p. 471, McGraw-Hill, 1981.

4. Marple S.L. Jr., *Digital Spectral Analysis with Applications*, Ch. 5, Prentice-Hall, Inc., 1987.

5. Go W.W., *The Use of Window Functions and Kalman Filtering in Spectral Estimation*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1990.

6. Chen C.H., *Signal Processing Handbook*, Ch. 4, Marcel Dekker, Inc., 1988.

7. Kay M.S., *Modern Spectral Estimation - Theory & Application*, Ch. 4, pp. 77-80, Prentice-Hall, Inc., 1988.

8. Zurbenko I.G., *The Spectral Analysis of Time Series*, Elsevier Science Publishers B.V., 1986.

9. Miller K.S. and Leskiw D.M., *An Introduction To KALMAN FILTERING WITH APPLICATIONS*, p. 1, Robert E. Krieger Publishing Company, Inc., 1987.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93940-5002	2
3.	Chairman, Code EC/Mw Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93940-5000	1
4.	Prof. Ralph Hippenstiel, Code EC/Hi Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93940-5000	3
5.	Prof. Roberto Cristi, Code EC/Cx Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93940-5000	1
6.	Prof. Monique Fargues, Code EC/Fa Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93940-5000	1
7.	Prof. Charles Therrien, Code EC/Th Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93940-5000	1
8.	LCDR Ronald C. Adamo, USN Executive Officer USS Brewton (FF-1086) FPO San Francisco, CA 96661-1446	2

9. Naval Ocean Systems Center 1
ATTN: Dr. C.E. Persons, Code 732
San Diego, CA 92152

10. Naval Ocean Systems Center 1
ATTN: Mr. Gary Thompson, Code 733
San Diego, CA 92152

Thesis
A2243 Adamo
c.1 Adaptive windows via
Kalman filtering in the
spectral domain.